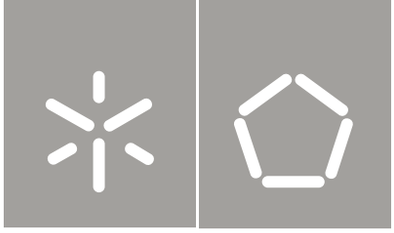
Universidade do Minho
Escola de Engenharia

Susana Isabel Pereira de Sousa

Encaminhamento de tráfego por Classes de Serviço em redes DiffServ

outubro de 2013

Encaminhamento de tráfego por Classes de Serviço em redes DiffServ
Susana Isabel Pereira de Sousa
UMinho | 2013



Universidade do Minho
Escola de Engenharia

Susana Isabel Pereira de Sousa

Encaminhamento de tráfego por Classes de Serviço em redes DiffServ

Tese de Mestrado
Ciclo de Estudos Integrados Conducentes ao
Grau de Mestre em Engenharia de Comunicações

Trabalho efetuado sob a orientação de
Professora Doutora Maria João Nicolau
Professor Doutor António Costa

outubro de 2013

DECLARAÇÃO

Nome: Susana Isabel Pereira de Sousa

Correio electrónico: susanamiecom@hotmail.com

Tlm.: 916840582

Número do Bilhete de Identidade:12506921

Título da dissertação:

Encaminhamento de Tráfego por Classes de Serviço em Redes DiffServ

Ano de conclusão: 2013

Orientadores:

Maria João Nicolau

António Costa

Designação do Mestrado:

Ciclo de Estudos Integrados Conducentes ao Grau de Mestre em Engenharia de Comunicações

Área de Especialização: Engenharia de Comunicações

Escola: Escola de Engenharia

Departamento: Sistemas de Informação/ALGORITMI

1. É AUTORIZADA A REPRODUÇÃO INTEGRAL DESTA DISSERTAÇÃO APENAS PARA EFEITOS DE INVESTIGAÇÃO, MEDIANTE DECLARAÇÃO ESCRITA DO INTERESSADO, QUE A TAL SE COMPROMETE.

2. É AUTORIZADA A REPRODUÇÃO PARCIAL DESTA DISSERTAÇÃO (indicar, caso tal seja necessário, nº máximo de páginas, ilustrações, gráficos, etc.), APENAS PARA EFEITOS DE INVESTIGAÇÃO, MEDIANTE DECLARAÇÃO ESCRITA DO INTERESSADO, QUE A TAL SE COMPROMETE.

3. De acordo com a legislação em vigor, não é permitida a reprodução de qualquer parte desta dissertação

Guimarães, ___/___/_____

Assinatura: _____

"Ninguém escapa ao sonho de voar,
de ultrapassar os limites do espaço onde nasceu,
de ver novos lugares e novas gentes.
Mas saber ver em cada coisa, em cada pessoa,
aquele algo que a define como especial,
um objecto singular, um amigo, - é fundamental.
Navegar é preciso, reconhecer o valor das coisas e das pessoas,
é mais preciso ainda"

Antoine de Saint-Exupéry

Agradecimentos

Esta dissertação representa o fim de um percurso académico longo e cheio de desafios. E porque o percurso que termina agora não foi solitário, eu quero deixar os meus mais sinceros agradecimentos a algumas pessoas.

Começo por agradecer aos meus orientadores, Professora Doutora Maria João Nicolau e Professor Doutor António costa, por todo o apoio e dedicação demonstrados no decorrer do trabalho, mas especialmente por ter sido com eles que descobri este gosto pelas redes de computadores.

Gostaria também de deixar uma palavra de apreço a todos os outros professores que me acompanharam ao longo da formação, porque todos, sem exceção, contribuíram para este desfecho positivo.

A todos os meus companheiros do curso, em especial a Adriana Costa, o João Galvão, o Pedro Freitas, o Pedro Queirós, e o Victor Magalhães, que apesar de muitas vezes eu ter desesperado, nunca me deixaram desistir de lutar.

Às minhas inestimáveis amigas, a Guerrinha, a Isaura, a Liliana, e a Raquel, que de uma forma ou de outra sempre me ajudaram a acreditar em mim.

Ao Zé, por ter partilhado tantos sentimentos comigo, e ter sido nos bons e nos maus momentos o meu porto de abrigo.

E por último à minha família, em especial à minha irmã e à minha mãe, que são as pessoas mais importantes da minha vida, e sempre me ajudaram sem olhar a meios.

A todos dedico este trabalho.

Resumo

Atualmente assistimos a um crescimento exponencial do número de aplicações que surgem para atender às necessidades dos utilizadores da *Internet*. As aplicações são cada vais mais avançadas e exigentes em termos de requisitos de *Qualidade de Serviço*. No entanto, os recursos e mecanismos disponibilizados pelas redes *IP* não acompanham esta evolução e não conseguem satisfazer estes novos requisitos. A comunidade científica tem proposto ao longo dos anos formas alternativas de resolver este problema, sendo uma delas a incorporação de mecanismos de *QoS* no encaminhamento. Neste sentido surgiram duas abordagens de encaminhamento com *Qualidade de Serviço*: uma abordagem por fluxo, onde as rotas são calculadas a pedido, e os recursos da rede são reservados ao longo de um caminho desde a origem até ao destino, para cada fluxo de dados; e uma abordagem por classes de serviço, onde os fluxos são agregados num número reduzido de classes de serviço, de acordo com os requisitos das aplicações, e os pacotes de dados são condicionados na rede *IP*, de acordo com a classe a que pertencem.

Esta dissertação propõe uma estratégia de encaminhamento por classes de serviço, com mecanismos de diferenciação nos encaminhadores da rede. Divide-se em dois componentes: um modelo de diferenciação baseado no funcionamento do *Diff-Serv*, ao nível dos encaminhadores; e um protocolo de encaminhamento *unicast* por classes de serviço, que resulta de uma extensão ao protocolo *OSPF*. Os pacotes de dados são marcados numa das classes de serviço, e tratados pelos encaminhadores da rede de acordo com a classe a que pertencem. O protocolo de encaminhamento, por sua vez, passa a incluir nas tabelas de encaminhamento os caminhos mais curtos baseados em métricas relacionadas com os requisitos das classes de serviço, sendo os pacotes de dados encaminhados pelo melhor caminho percecionado pela classe de serviço a que pertencem.

A estratégia foi implementada e testada usando o simulador de redes *Network Simulator 3*. O modelo de encaminhamento proposto foi alvo de comparação com outros modelos de encaminhamento e, mesmo numa situação de excesso de carga de tráfego com *QoS* a circular na rede, o modelo apresenta-se como uma solução eficiente para otimizar o desempenho das classes de serviço com requisitos de *QoS*, sem prejudicar o desempenho da classe *Best Effort*.

Palavras-Chave: *Internet*, *Qualidade de Serviço*, *Redes IP*, *Encaminhamento*, *Classes de Serviço*

Abstract

Currently we have seen an exponential growth in the number of applications that emerge to serve the needs of the *Internet* users. Applications are increasingly advanced and more demanding in terms of quality of service requirements. However, the resources and mechanisms provided by *IP Network* are not yet prepared to meet these new requirements. In attempt of solving this problematic, emerged two approaches of routing with quality of service, a per-flow approach, where routes are calculated on demand and network resources are reserved along a path from source to destination for each data flow; and an approach per-classe, where flows are aggregated into a small number of service classes, according to the requirements of the applications and data packets are conditioned in the *IP Network* in accordance with the class they belong to.

This Thesis presents a multi-classe routing strategy, with differentiation mechanisms in network routers. It is divided into two components: a model of differentiation based on *DiffServ*, and a unicast multi-classe routing protocol, based in an extension of *OSPF*. On the network routers, data packets are marked on one of the classes of service, and conditioned in accordance with the markings. The routing protocol in turn, includes in the routing tables the shortest paths based on metrics related to the requirements of classes of service, and data packets are forwarded by the best path according the class of service they belong.

The strategy was implemented and tested using *Network Simulator 3*. The routing model proposed was compared with other routing models, and even in a situation of excess of *QoS* traffic on the network, the model presented, appears as an efficient solution to optimize the performance of *QoS* traffic, without significantly impair the performance of the *Best Effort* traffic.

Keywords: *Internet*, *Quality of Service*, *IP Networks*, *Routing*, *Classes of Service*

Conteúdo

Lista de Figuras

Lista de Tabelas

Listings

Lista de Acrónimos

Acrónimo	Significado
AF	**A**ssured **F**orwarding
API	**A**pplication **P**rogramming **I**nterface
AS	**A**utonomous **S**ystem
BE	**B**est **E**ffort
bps	**b**its **p**or **s**egundo
CBR	**C**onstante **B**it **R**ate
CBS	**C**ommitted **B**urst **S**ize
CIR	**C**ommitted **I**nformation **R**ate
DS	**D**ifferentiated **S**ervices
DSCP	**D**ifferentiated **S**ervices **C**ode **P**oint
DVMRP	**D**istance **V**ector **M**ulticast **R**outing **P**rotocol
EBS	**E**xcess **B**urst **S**ize
EF	**E**xpedited **F**orwarding
FIFO	**F**irst **I**n **F**irst **O**ut
FTP	**F**ile **T**ransfer **P**rotocol
GCC	**G**NU **C**ompiler **C**ollection
HTTP	**H**ypertext **T**ransfer **P**rotocol
IP	**I**nternet **P**rotocol
ISP	**I**nternet **S**ervice **P**rovider
IETF	**I**nternet **E**ngineerin **T**ask **F**orce
LSA	**L**ink **S**tate **A**dvertisements
MOSPF	**M**ulticast **O**pen **S**hortest **P**ath **F**irst
MPLS	**M**ulti **P**rotocol **L**abel **S**withing
NS-3	**N**etwork **S**imulator **3**
NS-2	**N**etwork **S**imulator **2**
OSI	**O**pen **S**ystems **I**nterconnection
OSPF	**O**pen **S**hortest **P**ath **F**irst
PERD	**P**er-**C**lasse **D**issemination
PHB	**P**er **H**op **B**ehavior
PIR	**P**eak **I**nformation **R**ate
PQ	**P**riority **Q**ueue
QoS	**Q**uality **o**f **S**ervice
QRS	**Q**oS **R**outing **S**imulator
RIP	**R**outing **I**nformation **P**rotocol

Acrónimo	Significado
RSVP	Resource **Re**e**S**er**V**ation **P**rotocol
SLA	**S**ervice **L**evel **A**greement
SP	**S**hortest **P**ath
srTCM	single rate **T**ree **C**olor **M**eter
TC	**T**raffic **C**lass
TCP	**T**ransmission **C**ontrol **P**rotocol
ToS	**T**ype **o**f **S**ervice
trTCM	two rate **T**ree **C**olor **M**eter
VoIP	**V**oice **o**ver **I**nternet **P**rotocol
WB	**W**idest **B**andwith
WRED	**W**eighted **R**andom **E**arly **D**etection
WRR	**W**eighted **R**ound **R**obin

Capítulo 1

Introdução

1.1 Enquadramento

A *Internet* tem vindo a desempenhar um papel de grande relevo na sociedade atual. Este facto leva naturalmente ao aparecimento de novas aplicações para atender às necessidades dos utilizadores, aplicações essas cada vez mais avançadas e mais exigentes em termos de largura de banda. No entanto, os serviços disponibilizados pelas redes *IP* (Internet Protocol [2]) não são ainda adequados para atender a estas novas exigências, sendo que a solução não passa por, simplesmente, acrescentar largura de banda às redes. Em alternativa, devem ser disponibilizados mecanismos de Qualidade de Serviço (*QoS* - Quality of Service), de forma a que os fluxos de cada aplicação passem a ser "tratados"pela rede *IP* de acordo com os seus requisitos específicos. O serviço deixa de ser apenas de melhor esforço (*BE* - Best Effort), onde os pacotes de dados são encaminhados pelos routers tão rapidamente quanto possível, sem garantias relativas a atrasos, ordem de chegada e até mesmo de entrega ao destinatário. Dotar as redes *IP* de mecanismos de *QoS* implica a resolução de problemas de caráter multidimensional, isto pelo facto de existir uma gama alargada de soluções possíveis para os diferentes níveis da pilha protocolar e planos envolvidos (dados, controlo e gestão). O foco desta dissertação é a camada 3 da pilha, o nível da Rede. Tal como nas outras camadas da pilha, a implementação de mecanismos de *QoS* ao nível da rede é um desafio. É possível desenvolver mecanismos para trabalhar no plano de dados e de gestão, como classificação, marcação e policiamento de pacotes, controlo de congestionamento ou gestão de filas de espera, bem como no plano de controlo, como o controlo de admissão, reserva e atribuição de recursos ou encaminhamento de tráfego com restrições (*QoS* Routing). No *QoS* Routing o encaminhamento de tráfego é alterado de forma a considerar os requisitos de *QoS* das aplicações. O objetivo é garantir que as aplicações mais exi-

gentes disponham dos recursos necessários, ou pelo menos mais recursos do que as aplicações menos exigentes. Existem basicamente duas abordagens distintas quando se trata de encaminhamento de tráfego com *QoS*, por fluxo ou por classes de serviço. A abordagem de encaminhamento por fluxo parte do princípio que as rotas são calculadas a pedido. Cada pedido de encaminhamento de um novo fluxo inclui explicitamente as necessidades de recursos ao longo do percurso e cabe ao processo de encaminhamento encontrar um caminho que satisfaça essas necessidades. Caso exista, e se for encontrado, procede-se então a uma reserva de recursos ao longo do percurso encontrado, de modo a que as condições se possam manter até que o fluxo termine sem degradação. Essa reserva obriga à manutenção de informação de estado, por cada fluxo, em cada router. Por esse motivo, esta primeira abordagem sofre de problemas óbvios de escalabilidade. A abordagem alternativa é exclusivamente baseada em classes de serviço. Em vez de procurar satisfazer os requisitos de cada fluxo individualmente, a ideia é agrupá-los a todos num número muito reduzido de classes de serviço, para as quais se calculam os melhores caminhos, tendo em conta os requisitos pré-estabelecidos para cada classe. Os pacotes de cada fluxo são primeiramente marcados numa das classes de serviço, recebendo posteriormente o tratamento adequado a essa classe em todas as operações de encaminhamento ao longo do seu percurso na rede. A mudança de foco - de fluxo para classe - constitui na realidade uma importante mudança de paradigma. Os fluxos são agregados por afinidade em termos de requisitos de *QoS*, não sendo por isso possível dar garantias rigorosas a cada um. No entanto, e uma vez que temos um número reduzido de classes, é possível pré-calcular as rotas por cada classe em vez do cálculo a pedido. Dado que as rotas de cada classe podem diferir das restantes, existe aqui um enorme potencial para engenharia de tráfego e um reforço na diferenciação das classes pela via do encaminhamento. No âmbito desta dissertação será desenvolvido um trabalho, de cariz teórico e prático, focado na abordagem de diferenciação de tráfego ao nível do encaminhamento, para fornecer qualidade de serviço na *Internet*.

1.2 Objetivos

Como já foi referido, com este trabalho pretende-se estudar e avaliar formas de implementar o encaminhamento de tráfego por classes de serviço na *Internet*. Naturalmente que a abordagem por classes deve ser aplicada coerentemente, não só no tráfego *unicast* (de uma origem para um destino) como também no tráfego *multicast* (de uma origem para múltiplos destinos), que serve de suporte à maioria das aplicações colaborativas multimédia. E também no encaminhamento entre domínios, em larga escala, numa perspetiva fim-a-fim. Constituindo-se como uma mudança de paradigma, obriga naturalmente a reequacionar os processos de encaminhamento. O principal objetivo deste trabalho é o desenvolvimento de uma estratégia de encaminhamento de tráfego *unicast* por classes de serviço ao nível do inter-domínio. O desenvolvimento da estratégia passou pelas seguintes etapas:

- Estudar e compreender o funcionamento dos modelos atuais de encaminhamento de tráfego com QoS por classes de serviço, levantamento de vantagens e limitações dos mesmos;
- Conceber uma nova estratégia de encaminhamento por classes de serviço, baseada nos mecanismos e modelos de encaminhamento atuais;
- Definida a estratégia a adotar, determinar o suporte existente para os mecanismos e modelos por parte dos ambientes de simulação de redes IP;
- Implementação da estratégia proposta, recorrendo a um ambiente de simulação adequado;
- Desenvolver cenários de teste onde será feita uma avaliação da estratégia;
- Comparar a estratégia proposta com as soluções já existentes.

1.3 Estrutura do Documento

Este documento foi organizado em 7 capítulos. No primeiro capítulo é feito o enquadramento do trabalho desenvolvido na dissertação, onde é abordado o panorama atual e são identificados alguns dos desafios na implementação de um protocolo de encaminhamento por classes. São de seguida identificados os principais objetivos a serem alcançados e, por último, é apresentada a estrutura do documento.

No segundo capítulo são apresentado os conceitos relativos ao tema de qualidade de serviço na *Internet*. É aqui feita uma distinção entre a qualidade de serviço da perspetiva do utilizador e da perspetiva da rede. Sao apresentados os modelos normalizados de qualidade de serviço para redes *IP*.

No terceiro capítulo é apresentado o conceito de encaminhamento com qualidade de serviço. É caso de estudo o funcionamento de alguns protocolos de encaminhamento, e apresentados trabalhos realizados na área do encaminhamento com qualidade de serviço.

No quarto capítulo é apresentada uma proposta de encaminhamento com classes de serviço. São apresentadas as principais considerações e decisões tomadas para habilitar o encaminhamento de tráfego por classes de serviço.

No quinto capítulo é éfetuada a descrição da implementação do sistema utilizando o simulador de redes *Network Simulator 3*. São descritas as funcionalidades do simulador e as alterações efetuadas para habilitar o encaminhamento por classes de serviço. Neste capítulo é também descrito o modelo de diferenciação que foi integrado com o simulador, e as principais alterações efetuadas para habilitar a diferenciação nos encaminhadores.

No sexto capítulo é apresentada a validação da implementação de encaminhamento por classes de serviço. São retratados 4 cenários de encaminhamento distintos e feita uma análise comparativa entre eles, no sentido de determinar as vantagens e desvantagens de cada um deles.

O sétimo capítulo apresenta as conclusões obtidas no decorrer do trabalho desenvolvido. São identificados os objetivos alcançados e efetuadas algumas considerações para trabalho futuro.

Capítulo 2

Qualidade de Serviço em Redes IP

2.1 Qualidade de Serviço

A *Internet* é largamente utilizada para comunicação de dados de todo o tipo de serviços e aplicações. Atualmente assistimos a um crescimento exponencial do tráfego de dados que circula na rede, facto que influencia negativamente o desempenho de serviços ou aplicações, mais exigentes em termos de requisitos de funcionamento. As aplicações de tempo real, em que os fluxos de dados são muito sensíveis a variações de diversas ordens, como largura de banda, atraso, etc., são um exemplo. Os requisitos variam de acordo com o tipo de aplicação, e naturalmente das expetativas do utilizador, e é responsabilidade da rede assegurar a disponibilidade dos recursos que satisfaçam as necessidades, de forma diferenciada, de acordo com a aplicação. Ou seja, a rede deve garantir Qualidade de Serviço[3].

A *Internet* não foi, no entanto, inicialmente projetada para lidar com este tipo de tráfego gerado pelas aplicações emergentes [4]. A Rede *IP* foi desenvolvida nos anos 70 tendo em conta um único requisito, o de conectividade. Oferece um serviço de *datagrama* não orientado à conexão, sem garantias de entrega, não podendo naturalmente ser feitas quaisquer assunções relativas a largura de banda, atrasos, variação de atrasos ou perda de *datagramas*. Desta forma tornou-se necessário a introdução de mecanismos nas diferentes camadas da pilha de comunicações para dotar a rede de qualidade de serviço.

O modelo de camadas da *Internet*, modelo *TCP/IP*, concebido a partir do modelo tradicional *OSI*, é constituído por 5 níveis que contêm os protocolos sobre os quais as aplicações que circulam na rede *Internet* funcionam (Figura 2.1).

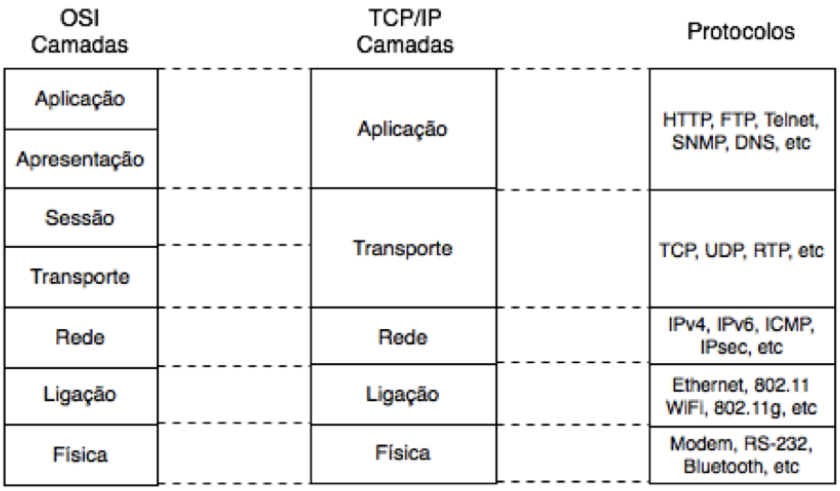


Figura 2.1: Modelo TCP

Os mecanismos de *QoS* são passíveis de ser implementados em qualquer nível da pilha, mas o foco do presente trabalho será o *QoS* ao nível da rede, nomeadamente no protocolo *IP*. É necessário adaptar o comportamento dos encaminhadores, para que fluxos com necessidades mais exigentes de *QoS* recebam algum tipo de tratamento preferencial, comparado com os fluxos de dados menos sensíveis. É também necessário influenciar o processo de encaminhamento de forma a direcionar os fluxos de dados pelos caminhos que melhor satisfaçam os requisitos das aplicações (Figura 2.2).

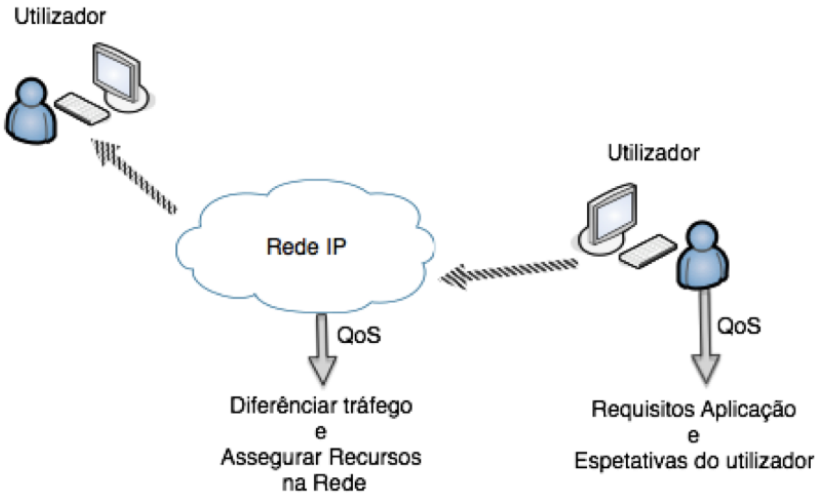


Figura 2.2: Perspectivas de Qualidade de Serviço [1]

2.1.1 Requisitos das Aplicações

Tal como referido anteriormente, cada vez mais têm surgido aplicações distintas, que geram fluxos de tráfego com requisitos de serviço diferentes, e que vão circular por toda a rede. Quando pretendemos diferenciar o tratamento dado ao tráfego que circula na rede, primeiramente devemos proceder à sua caraterização. É necessário determinar em que medida uma aplicação tolera ou não determinado comportamento da rede. Para tal vamos classificar as aplicações em dois grupos, as *Aplicações Elásticas* e as *Aplicações Inelásticas*[5].

As *Aplicações Elásticas* caraterizam-se por ter uma boa tolerância à variação do atraso e taxas de transmissão. As aplicações inseridas neste grupo têm um excelente desempenho com baixos níveis de atraso, mas não são completamente afetadas quando o atraso aumenta. A taxa de transferência pode ou não ser um problema, dependendo da natureza da aplicação em concreto. Exemplos de aplicações elásticas são as de *transferência de ficheiros*, *E-mail*, *FTP*, ou *HTTP*, onde a única preocupação se prende efetivamente com a integridade dos dados gerados pelas aplicações, preocupação contornada com o uso de protocolos de transporte adequados.

Por sua vez as *Aplicações Inelásticas* são geradoras de tráfego muito sensível a atraso, variação de atraso, taxa de transmissão e perda de pacotes. Neste grupo inserem-se na sua maioria as aplicações tempo-real, algumas mais tolerantes, como por exemplo Streaming de *Voz* ou *Video*, que têm a capacidade de se adaptar a variações de atrasos ou taxas de transferência, ajustando as caraterísticas do tráfego de acordo com o estado da rede. No entanto, muitas aplicações de tempo-real são rígidas em termos de requisitos e na sua maioria têm uma tolerância muito baixa a qualquer tipo de variação. Exemplo deste tipo de aplicações inelásticas com requisitos rígidos são as chamadas de voz sobre IP (VoIP), ou chamadas de *Video-conferência*, que têm uma tolerância muito baixa ao atraso e à variação do atraso, no sentido de não afetar a qualidade da sessão de comunicação.

2.1.2 Parâmetros de QoS

No processo de categorização de aplicações geradoras de tráfego surgem sempre conceitos como atraso, variação do atraso, perda de pacotes ou taxa de transferência. Estes conceitos representam, na prática, a resposta da rede ao tráfego que nela circula, daí serem designados de parâmetros de *QoS*. Fazendo uma avaliação destes parâmetros à medida que o tráfego circula na rede, podemos estimar um nível de

desempenho para as aplicações, com mais ou menos garantias dependendo da natureza da aplicação.

Os parâmetros de *QoS*, tipicamente usados para avaliar a performance de uma rede *IP* são, o *Atraso*, do inglês *Delay*, que corresponde ao tempo total que um pacote demora a percorrer um percurso entre dois pontos na rede (fim a fim, da origem ao destino, ou entre dois encaminhadores). Vários fatores influenciam o *Atraso*, desde o tempo de transmissão e propagação, ou o tempo de processamento e espera nas filas dos encaminhadores. Por esse facto, tipicamente não é possível dar garantias rígidas a uma aplicação de que o atraso se vai manter constante durante todo o percurso, para todos os pacotes de dados. Pode-se no entanto, mais uma vez dependendo da natureza da aplicação, definir com alguma garantia um atraso máximo ou mínimo, que é respeitado durante o serviço.

Outro parâmetro chave na avaliação do desempenho das aplicações é uma derivação do *Atraso*, e corresponde à variação do mesmo, do inglês *Jitter*. A *Variação do Atraso* representa a diferença de atrasos entre pacotes de um mesmo fluxo, ou seja, o tempo que um pacote demora a percorrer determinado percurso (*Atraso*) não é necessariamente o mesmo para os restantes pacotes do fluxo. O que implica muitas vezes que os pacotes cheguem aos destinatários não exatamente pela ordem com que foram gerados na origem, situação que, dependendo da aplicação, implica a utilização de mecanismos adicionais no processo de reordenação. A variação do atraso é provocada por diversos factores, como protocolos de encaminhamento, agregação de tráfegos, mecanismos de escalonamento das filas nos encaminhadores, etc. Relativamente a garantias que possam ser dadas a aplicações no processo de acordo de serviço, este parâmetro é bastante complexo de avaliar, tipicamente são acordados valores aproximados, baseados na diferença entre o atraso máximo e mínimo, ou determinados dinamicamente durante a transmissão do fluxo de dados pela rede.

O parâmetro *Perda de Pacotes*, do inglês *Packet Loss*, é também um parâmetro a considerar no desempenho da rede, e corresponde à quantidade de pacotes de dados de um fluxo que foram gerados mas não alcançaram o destinatário. A perda de um ou mais pacotes de dados pode ocorrer por congestão nos encaminhadores, por violação de limites de tráfego, falhas na rede, etc. Para aplicações de tráfego real a perda de pacotes pode ser crítica, porque não é possível aplicar um algoritmo de retransmissão de pacotes. As caraterísticas tão imprevisíveis deste parâmetro tornam muito difícil de dar garantias às aplicações. A perda de pacotes é tipicamente determinada em termos de probabilidade de ocorrência.

Por último a *Taxa de transmissão* numa rede *IP*. Este parâmetro pode variar em questão de segundos, horas ou dias e é o recurso mais relevante para a maioria das aplicações que correm sobre *IP*. Não existem garantias absolutas em relação à taxa de transmissão, e fatores como o protocolo de encaminhamento, a conetividade, a congestão e a agregação de tráfego contribuem para esse facto. Muitas aplicações produzem tráfego em rajadas, provocando picos de taxas de transmissão, o que dificulta muito quando se pretende especificar este parâmetro de forma a dar algumas garantias às aplicações. Dependendo da aplicação que requisita o serviço, podem ser pedidas garantias de taxa médias, de pico ou ambas.

2.2 Modelos de QoS em Redes IP

2.2.1 Modelo de Melhor Esforço

Em encaminhadores *IP*, o encaminhamento de *datagramas* é baseado em tabelas de encaminhamento construídas usando métricas simples e endereços de destino [6]. Não é efetuada nenhuma análise do tipo de tráfego que cada *datagrama* pode conter, sendo que todos os dados são tratados de igual forma, com a mesma prioridade.

Durante uma sessão de comunicação, os pacotes *IP* percorrem caminhos na rede, estabelecidos de acordo com entradas nas tabelas de encaminhamento. O único dinamismo estabelecido deve-se à robustez do protocolo no caso de ocorrência de falhas nas ligações ou sobrecarga no encaminhador. Este facto leva a que não exista um caminho fixo através da rede, não sendo possível garantir que a qualidade de serviço oferecida a determinado fluxo permaneça coerente durante toda a sessão de comunicação. E mesmo que o caminho se mantenha estável, não existem garantias de que os pacotes de um fluxo não possam afectar pacotes de outro fluxo distinto, já que existe uma partilha parcial ou mesmo total dos caminhos da rede.

Os encaminhadores têm como função analisar os pacotes que entram, de forma a determinar qual o caminho através do qual devem ser encaminhados; dependendo do destinatário é determinada a interface de saída correta. Os pacotes associados a cada interface de saída são mantidos numa fila de espera do tipo *FIFO*(o primeiro a entrar é o primeiro a sair), em que o pacote na frente da fila é o primeiro a ser transmitido. Este processo executado pelo encaminhador mostrou-se como um bom ponto de partida para a introdução de mecanismos de *QoS* ao nível da rede. Seria útil que nas filas dos encaminhadores, a tráfego com requisitos de *QoS* mais exigentes fosse atribuída uma prioridade superior à do tráfego com requisitos de *QoS* menos exigentes.

Os protocolos de encaminhamento dinâmico, utilizados para construir as tabelas de encaminhamento, escolhem as melhores rotas tendo em conta uma única métrica, tipicamente o caminho mais curto em número de saltos ou custo da ligação. O encaminhamento com *QoS* tem como objetivo otimizar o processo de seleção de rotas, utilizando outro tipo de métricas, individualmente ou combinadas, que vão de encontro aos requisitos dos fluxos de tráfego, nomeadamente:

- **Largura de Banda**
- **Atraso**
- **Variação do Atraso**
- **Perdas de pacotes**

No sentido de melhorar o serviço oferecido pelas redes *IP*, passando a considerar as caraterísticas do tráfego gerado pelas novas aplicações, bem como as expetativas dos utilizadores, surgiram alguns mecanismos de suporte ao *QoS* na rede. Mecanismos como policiamento e políticas de encaminhamento, escalonamento e gestão de filas nos encaminhadores, reserva de recursos ou classificação e marcação de pacotes.

2.2.2 Modelo de Serviços Integrados

O modelo de serviços integrados (*IntServ*), descrito em [7], trata-se do primeiro modelo normalizado, proposto pela *IETF* para dar suporte a *QoS* na *Internet*. A arquitetura *IntServ* permite definir como os elementos da rede devem tratar fluxos individuais de tráfego, através da reserva de recursos fim a fim, ainda antes do início da transmissão de dados.

O modelo baseia-se na ideia de circuito virtual, no sentido de que estabelece uma reserva de recursos fixa por onde os pacotes pertencentes a determinado fluxo de tráfego com requisitos de QoS são encaminhados. Os diferentes fluxos são definidos pelos pares endereço de origem/endereço de destino e porta de origem/porta de destino.

O tipo de serviço atribuído a um fluxo depende das duas especificações, nomeadamente dos requisitos de *QoS* e das caraterísticas do fluxo definidos em [8][9]. O modelo *IntServ*, além de *Best Effort*, define mais dois níveis de serviço, relacionados com os requisitos de *QoS*:

- **Serviço de Carga Controlada** do inglês *Controlled Load* Service [10], que fornece qualidade de serviço um pouco superior ao *Best Effort*, para aplicações tolerantes a atrasos e a perdas de pacotes de baixa ordem. O desempenho tende a degradar-se quando aumenta a carga na rede;

- **Serviço Assegurado** do inglês *Guaranteed Service* [11], que garante limites máximos nas taxas de atrasos, descartes de pacotes nas filas de espera e disponibilidade de largura de banda.

Quando uma aplicação pretende iniciar uma transmissão, efetua um pedido à rede pelo protocolo de sinalização *RSVP*, por sua vez a rede vai verificar se dispõe dos recursos necessários para satisfazer o pedido. Esta função da rede, designada de controlo ou chamada de admissão (do inglês *call admission*), além de determinar a viabilidade de determinado pedido, faz o policiamento durante o uso do serviço. Uma vez uma chamada de serviço aceite, a rede deve despoletar mecanismos de escalonamento e gestão de filas, de forma a garantir que os fluxos de dados recebam o tratamento solicitado. A semântica do serviço deve ser do conhecimento dos encaminhadores e também das aplicações que vão ejetar tráfego no domínio *IntServ*.

O *RSVP* é um protocolo de sinalização da camada de rede, que permite o envio de mensagens periódicas pelo emissor, designadas *PATH*, com as especificações dos fluxos de tráfego (FlowSpec - Flow Specification), como os requisitos de *QoS* e características do fluxo. O recetor, por sua vez, solicita a reserva de recursos com o envio de mensagens *RESV*, pelo caminho inverso do das mensagens *PATH*. A mensagem *RESV* inclui a especificação dos recursos (RSpec - Resource Specification), com o nível de serviço atribuído ao fluxo e outra informação para configurar o policiamento nos encaminhadores.

Os elementos *IntServ* da rede, quando é viável, procedem à reserva de recursos, de forma unidirecional, ao longo de um caminho para tráfego *unicast*, ou ao longo de uma árvore para tráfego *multicast*. O protocolo é suportado para a versão 4 do protocolo da *Internet* (*IPv4*), pelo campo do cabeçalho *Type-of-Service* [12], e para *IPv6*, pelo campo *Flow Label* [13].

A reserva de recursos apresenta no entanto bastantes problemas, o principal é o facto de que os recursos na rede *IP* são bastante limitados, fazendo com que muitos pedidos de reserva sejam recusados. Por outro lado, mesmo que se efetive a reserva, basta uma simples falha de comunicação para voltarmos ao serviço de melhor esforço, e desperdiçar recursos preciosos durante o processo de nova reserva.

Todos estes motivos contribuem para que este modelo não seja ainda aplicado a

uma escala global, sendo apenas usado em ambientes de redes controlados.

2.2.3 Modelo de Serviços Diferenciados - DiffServ

O modelo *DiffServ* do *IETF*, descrito em vários documentos [14][15][16], surgiu como alternativa ao modelo *IntServ*, e apresenta-se como uma solução bastante mais flexível e escalável para implementar o *QoS* ao nível da rede. De acordo com este modelo os fluxos de tráfego são agregados em classes de serviço com diferentes prioridades, sendo o encaminhamento diferenciado por classe e não por fluxo, garantindo uma maior escalabilidade na gestão dos recursos da rede. Os pacotes de dados recebem uma marcação, de acordo com requisitos específicos, e são processados e diferenciados pelos elementos da rede *DiffServ*. A marcação define a classe a que o pacote pertence, e pode tratar-se da classe de mais baixa prioridade, designada de *Best-Effort* (BE), ou de uma das classes de alta prioridade, classes de *QoS* do modelo.

De salientar que, apesar de garantir recursos a fluxos de tráfego mediante a marcação e priorização por classes, o *DiffServ* não define um serviço fim a fim, apenas comportamentos de *QoS* nos encaminhadores da rede, e não na forma como o tráfego é encaminhado. Passa a existir uma diferenciação do tráfego nas filas, mas o caminho selecionado para o fluxo percorrer a rede é independente do tratamento dado à classe no encaminhador.

A arquitetura *DiffServ* consiste num conjunto de elementos funcionais, implementados nos encaminhadores da rede que constitui um *Domínio DiffServ* (DS Domain)(Figura 2.3).

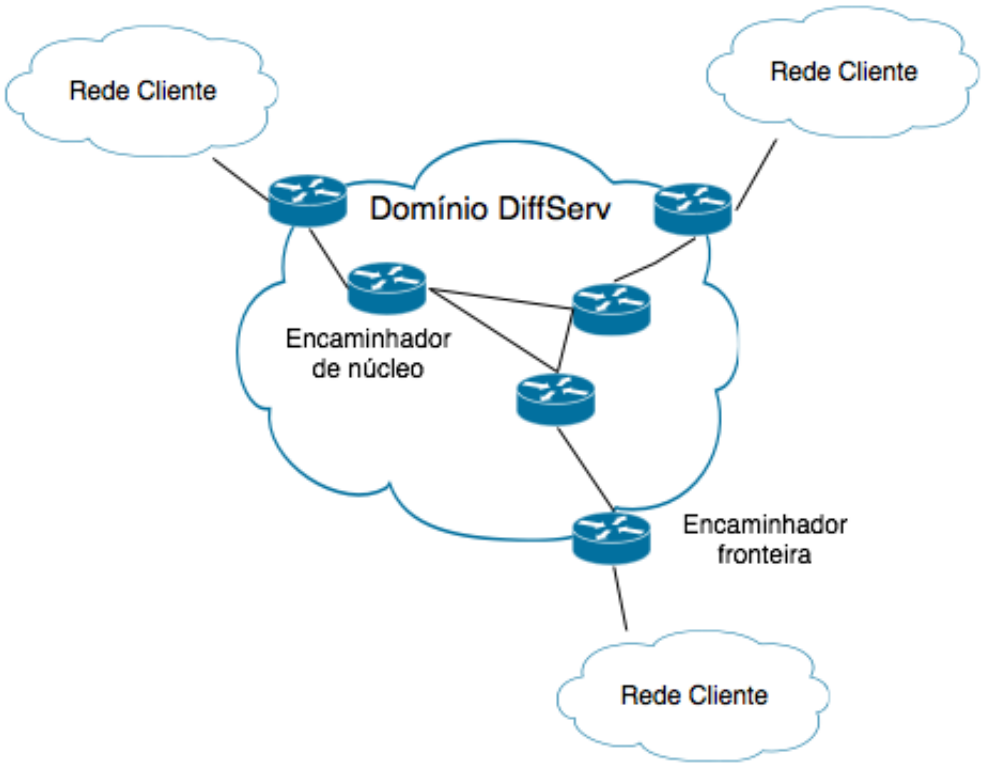


Figura 2.3: Modelo DiffServ

Existem dois tipos de encaminhadores num domínio *Diffserv*, os *encaminhadores de fronteira* (edge nodes), e os *encaminhadores internos* (core nodes), cada um com funções bem definidas no modelo:

Os **encaminhadores de fronteira** fazem toda a classificação, marcação e condicionamento do tráfego, além de estabelecerem a conetividade do domínio com outros domínios, *DiffServ* ou não.

Os **encaminhadores internos** estabelecem conetividade com os encaminhadores dentro do domínio e fazem a alocação de recursos de forma diferenciada de acordo com a classe de serviço a que os pacotes pertencem.

Acordo de nível de Serviço

Para que o tráfego gerado por uma aplicação seja alvo de diferenciação, no núcleo de um domínio *DiffServ*, numa primeira fase o cliente deve estabelecer um acordo com o *provedor de serviços da Internet* (ISP), na forma de *SLA*(Service Level Agre-

ements). Os *SLAs* especificam detalhadamente o tipo de requisitos do tráfego, bem como a forma como este deve ser processado e classificado dentro do domínio.

Depois de estabelecido o acordo, todos os pacotes do fluxo de dados são marcados, no campo *DS* (Differentiated Services) do cabeçalho *IP*, com a classe a que pertencem, o que corresponde na prática à atribuição do *DSCP*(Differentiated Services Code Point). Cada classe de serviço tem associádo um valor de *DSCP* único, com um nível de precedência e descarte normalizado[17].

Como podemos observar na Figura 2.4, o campo *DS* utiliza o campo *ToS*, na versão 4 do protocolo *IP* (IPv4), e o campo *Traffic Class octec* na versão 6 (IPv6)[15].

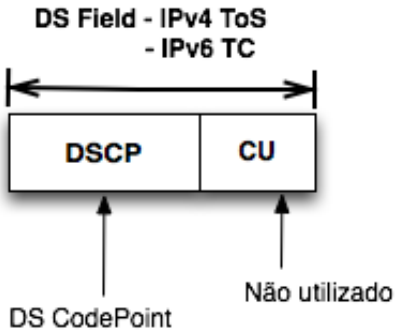


Figura 2.4: DSCP

Classificação, Marcação e Condicionamento

O *DSCP* é usado, à entrada do domínio pelos encaminhadores fronteira, para identificar que tipo de serviço deve ser fornecido ao pacote. O que significa que pacotes com o mesmo *DSCP* recebem o mesmo tipo de tratamento por parte dos encaminhadores. Este tratamento, designado no modelo de *PHB* (Per-Hop Behavior), define as políticas de priorização e descarte aplicadas aos pacotes, nas filas do encaminhador. Os principais *PHBs* normalizados para a arquitetura *Diffserv*, além do *Best Effort* são:

- **AS**(Assured Forward)[18] - Oferece um serviço melhor do que o *Best Effort*, no entanto sem garantias rígidas de *QoS*, pelo facto de não estabelecer limites superiores para atraso e variação do atraso. Nesta classe, o tráfego é dividido em 4 subclasses, cada uma com 3 níveis de precedência de descarte e espaço de armazenamento mínimo garantido nas filas dos dispositivos de rede.

- **EF**(Expedited Forwarded)[19] - Oferece garantias absolutas em termos de atraso, perdas e variação do atraso, além de largura de banda fixa entre duas aplicações. Deve ter-se muita atenção na alocação de recursos para o tráfego pertencente a esta classe, já que num caso limite pode anular por completo o desempenho das restantes classes.

Depois de classificado, o pacote passa por um processo de condicionamento, onde é feita uma comparação entre os valores acordados no estabelecimento do *LSA*, no sentido de determinar em que medida o pacote se encontra em conformidade com os mesmos. Este processo define se o pacote vai manter a marcação original ou, no caso de não cumprir o perfil acordado, se vai receber nova marcação ou ser descartado. Receber nova marcação tipicamente significa ser escalonado para uma fila de encaminhamento com menor prioridade. Na Figura 2.5 podemos visualizar os elementos funcionais que constituem os encaminhadores fronteira.

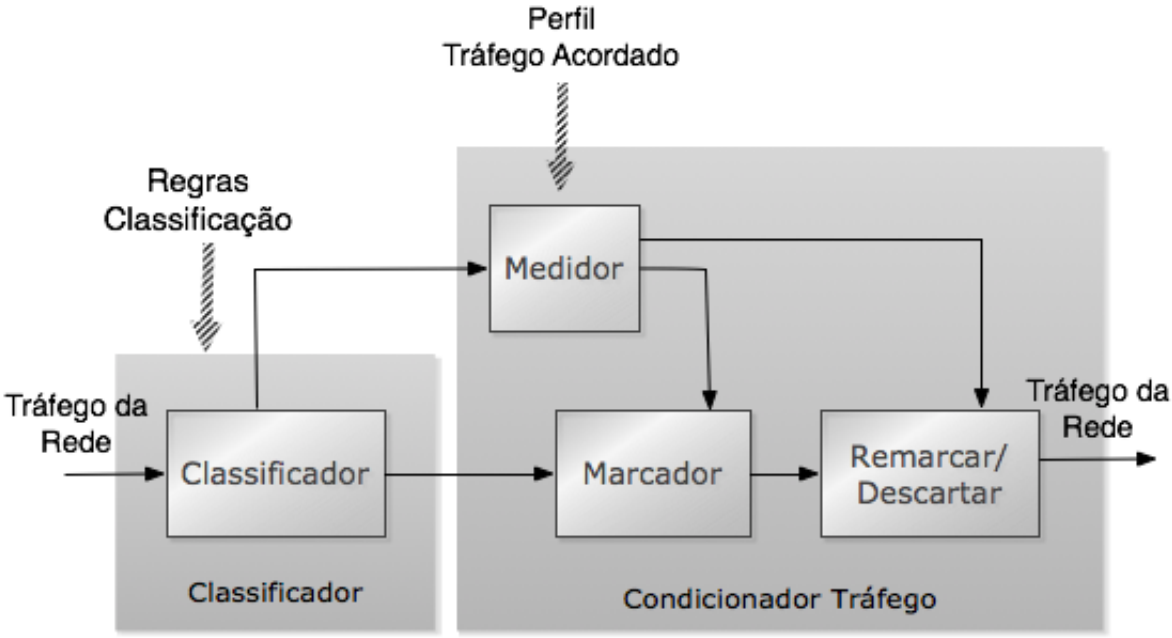


Figura 2.5: Elementos Funcionais DiffServ

Mecanismos de policiamento

Um dos parâmetros acordados com o provedor de serviços é o algoritmo a ser utilizado pelos encaminhadores de fronteira no processo de medição. O medidor vai determinar o nível de conformidade do pacote com o perfil de tráfego acordado no *SLA*.

Existem alguns algoritmos passíveis de serem implementados nos encaminhadores para o efeito, sendo que os que se apresentam de seguida serão caso de estudo neste trabalho prático.

Single Token Bucket Este algoritmo necessita que sejam estabelecidos dois parâmetros:

- **CIR**(Committed information Rate) - Taxa de informação que representa a periodicidade com que são gerados os *tokens* (em *Bytes*).
- **CBS**(Committed Burst Size) - Tamanho do *burst*, que representa o número total de *tokens* que é possível gerar, sendo que acima deste limite são descartados.

Quando um pacote entra neste bloco de medição, é feita uma comparação entre o tamanho do pacote e o número de *tokens* gerados, de forma a determinar se este se encontra em conformidade com o perfil. Se o tamanho do pacote for igual ou inferior ao número de *tokens*, significa que se encontra em conformidade com o perfil traçado.

srTCM (Single Rate Three-color Marker) Este algoritmo acrescenta mais um parâmetro aos já especificados no TokenBucket:

- **EBS**(Excess Burst Size) - O número de *tokens* extra que é possível gerar.

Tal como no algoritmo anterior, os *tokens* vão sendo gerados de acordo com o *CIR*, e quando atingem o limite de armazenamento (*CBS*) passam a ser armazenados na zona de excesso até atingirem o *EBS*. Quando chega um pacote é feita a comparação do tamanho com o número de *tokens* armazenados. Se o número de *tokens* for superior ou igual ao tamanho do pacote, este é considerado em conformidade verde. Caso os *tokens* não sejam suficientes, mas somando os *tokens* em excesso, se chegue ao tamanho do pacote, então o pacote é considerado em não-conformidade amarela. No limite, quando o tamanho do pacote é superior à totalidade dos *tokens*, então é considerado em não-conformidade vermelha.

trTCM (Two Rate Three-color Marker) Este algoritmo especifica mais um parâmetro além de *CIR*, *CBS* e *EBS*:

- **PIR**(Peak Information Rate) - Taxa de informação que representa a periodicidade com que são gerados *tokens* para uma nova zona de armazenamento.

Neste algoritmo são simultaneamente gerados *tokens* com periodicidade *PIR* e *CIR*. Quando chega um pacote é feita uma comparação entre o seu tamanho e o número de *tokens* gerados com a nova taxa de pico *PIR*, se o numero de *tokens* não for suficiente o pacote é considerado em não-conformidade vermelha. Se o número de *tokens* for suficiente é feita nova comparação, agora com os *tokens* gerados com periodicidade *CIR*, se o número de *tokens* não for suficiente o pacote é considerado como não-conforme amarelo. Se nas duas comparações o número de *tokens* for suficiente, o pacote apresenta uma conformidade verde.

As ações a tomar depois da medição (especificadas no *SLA*) vão depender do algoritmo aplicado e do nível de não-conformidade, sendo o pacote descartado ou remarcado com um *DSCP* de uma classe menos prioritária.

No núcleo do domínio, nos encaminhadores internoses, tal com referido anteriormente, é aplicado o *Per-Hop-Behavior* às classes de tráfego (Figura 2.6).

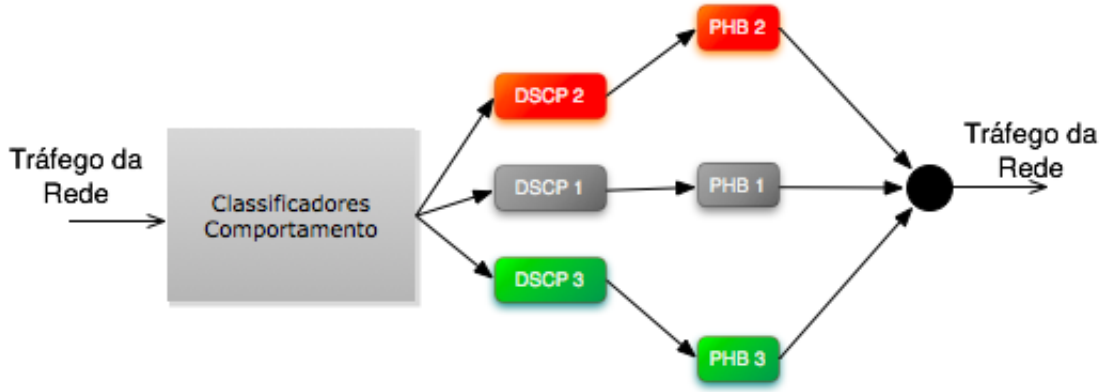


Figura 2.6: Classificador

Mecanismos de Escalonamento e Gestão de Filas

Para uma melhor compreensão do que acontece no núcleo de um domínio de diferenciação, vão ser abordados os principais algoritmos de escalonamento e gestão de filas, implementados nos encaminhadores do modelo *DiffServ*. Os algoritmos são aplicados com o propósito de estabelecer os diferentes níveis de prioridade e precedência de descarte. A sua utilização pode variar de implementação para implementação, sendo que o fundamental é prestar um serviço adequado às aplicações, tendo em consideração os recursos disponíveis na rede.

Os algoritmos de escalonamento de pacotes são responsáveis por selecionar pacotes das filas para que sejam transmitidos na rede. São três os algoritmos de escalonamento mais usados no modelo *DiffServ*:

FIFO (First In First Out) Este é o algoritmo de escalonamento mais simples do modelo e dos mais utilizados nas redes atuais. Os pacotes de dados vão sendo retirados das filas dos encaminhadores, pela mesma ordem que lá são colocados, ou seja, o primeiro a entrar é o primeiro a sair. É aplicado em todas as filas do modelo, desde a fila de menos prioridade, *BE* até à fila de maior prioridade *EF*.

WRR (Weighted Round Robin) Algoritmo aplicado quando combinamos o escalonamento de múltiplas filas. De forma sequencial e ordenada o escalonador retira um pacote de cada fila de diferenciação. De acordo com o peso definido para a fila, é retirado um ou mais pacotes de cada vez. No modelo *DiffServ* este algoritmo normalmente combina apenas o escalonamento das filas *AF* e *BE*, sendo que tal como referido anteriormente a fila *EF* tem precedência absoluta relativamente às restantes.

PQ (Priority Queue) Este algoritmo de escalonamento também seleciona os pacotes das filas dos encaminhadores, mas de acordo com prioridades. A fila mais prioritária é sempre servida em primeiro lugar, a não ser no caso de se encontrar vazia. No modelo este algoritmo é aplicado na combinação da fila de tráfego *EF* com as restantes filas de *QoS* e fila *BE*.

Além dos mecanismos de escalonamento, as filas dos encaminhadores no domínio de diferenciação estão também dotadas de mecanismos que fazem uma gestão adequada das mesmas. O espaço de armazenamento das filas tem limites e, como tal, é necessário definir que estratégia adotar no caso de se verificar a congestão da fila.

Seguidamente são descritos dois algoritmos de descarte de pacotes e controlo de congestão implementados no modelo.

DropTail Com o mecanismo *DropTail*, quando a fila do encaminhador atinge o limite, os pacotes que estão no fim da fila (na cauda), são descartados. Não existe qualquer tipo de diferenciação em relação aos pacotes que são descartados, e só entram novos pacotes na fila quando voltar a existir espaço de armazenamento. Este algoritmo é aplicado tipicamente às filas que recebem tráfego *BE*.

WRED (Weighted Random Early Detection) Trata-se de um mecanismo inteligente de descarte, que atua ainda antes de ocorrer a congestão. O tamanho das filas vai sendo monitorizado periodicamente, e de acordo com o resultado da monitorização, a probabilidade de descarte aumenta ou diminui em proporcionalidade. Normalmente este algoritmo é aplicado em filas de tráfego *AF*, que é caraterizado pela elevada sensibilidade a perdas de pacotes. O algoritmo tem políticas de descarte mais ou menos rigorosas, dependendo da prioridade da fila.

Capítulo 3

Encaminhamento com Qualidade de Serviço

3.1 Encaminhamento em redes IP - protocolo OSPF

A *Internet* engloba um conjunto de topologias simples de administrar, designadas de *Sistemas Autónomos*. Um sistema autónomo contém um identificador único e respeita um conjunto de políticas, no que respeita ao processo de encaminhamento, seja dentro do *sistema autónomo* (intra-domínio) ou entre sistemas distintos (inter-domínio).

O encaminhamento de tráfego em redes *IP* envolve a troca de informação de conetividade e acessibilidade entre os encaminhadores da rede, para que sejam construídas tabelas de encaminhamento. Com a informação os encaminhadores constroem tabelas com os caminhos para os destinos, sendo que cada entrada na tabela é uma associação entre um destino e o interface de saída por onde os pacotes devem ser encaminhados. Quando um pacote de dados chega a um encaminhador, este pesquisa na tabela qual o caminho para o destino especificado no cabeçalho *IP* do pacote, e encaminha-o pelo interface de saída correto. Este processo repete-se em todos os encaminhadores ao longo do caminho percorrido pelo pacote de dados, desde a origem até ao destino.

A construção das tabelas e o encaminhamento de pacotes é da responsabilidade do protocolo de encaminhamento implementado dentro de cada *sistema autónomo*, e a forma como tudo se processa varia de protocolo para protocolo.

Existem diversos protocolos de encaminhamento normalizados, como o *RIP* (Routing Information Protocol)[20] ou o protocolo *OSPF* (Open Short Path First) [21], para encaminhamento dinâmico de tráfego *unicast*, e o *DVMRP*(Distance Vec-

tor Multicast Routing Protocol)[22] ou o *MOSPF*(Multicast Open Shortest Path First)[23] para encaminhamento de tráfego *multicast*. Neste trabalho apenas será abordado o funcionamento do *OSPF* intra-domínio, já que se trata do protocolo de encaminhamento utilizado na proposta de encaminhamento por classes de serviço.

OSPF- (Open Shortest Path First)

Trata-se de um protocolo de encaminhamento de caminho mais curto, baseado no estado das ligações. Foi desenvolvido pelo *IETF* [21] para redes *IP* e é o protocolo de encaminhamento global mais amplamente utilizado nas redes de computadores reais, estando integrado na maioria dos encaminhadores.

Uma das principais caraterísticas do *OSPF* é o facto de que os encaminhadores dentro do *Sistema Autónomo* se poderem subdividir em grupos mais pequenos, designados *áreas*, e cada encaminhador mantém, numa base de dados topológica, apenas a informação de encaminhamento da área de rede onde está conetado, reduzindo desta forma o tamanho das tabelas e consequentemente o custo de processamento.

Neste protocolo a informação flui entre os encaminhadores da rede na forma de *LSAs* (Link State Advertisements), anúncios periódicos com o estado das ligações, e a métrica utilizada no protocolo para representar o custo da ligação está relacionada tipicamente com a *largura de banda*. Com a informação dos *LSAs* cada encaminhador calcula os caminhos mais curtos para os destinos aplicando um algoritmo de caminho mais curto *SPF* (Shortest Path First). O *OSPF* permite ainda a definição de mais que um caminho para o mesmo destino, no sentido de balancear a carga de tráfego na rede.

A construção das tabelas de encaminhamento processa-se nas seguintes etapas:

- Periodicamente cada encaminhador testa as suas ligações com os encaminhadores vizinhos, tipicamente a cada 10 segundos, utilizando o protocolo *Hello*. O protocolo *Hello* é responsável pela troca de mensagens entre encaminhadores vizinhos, no sentido de estabelecer e manter ativas as ligações entre eles.

- O estado das ligações é guardado num *LSA* e posteriormente enviado para os outros encaminhadores da rede num processo de inundação pelo protocolo *Flooding*, sendo este responsável por anunciar alterações na topologia.

- Cada encaminhador constrói, numa fase inicial, uma base de dados topológica, através da sincronização com as bases de dados dos encaminhadores adjacentes, pelo protocolo *Exchange*. O protocolo *Exchange* sincroniza encaminhadores

vizinhos recém descobertos. As bases de dados topológicas são idênticas em todos os encaminhadores da mesma área .

- Posteriormente, com a troca dos *LSAs*, são contruídas as bases de dados topológicas, idênticas em encaminhadores da mesma área *OSPF*.
- Os encaminhadores procedem ao cálculo dos caminhos mais curtos para cada destino, utilizando o algoritmo de cálculo de caminho mais curto *SPF* baseado no algoritmo de *Dijkstra*. O algoritmo de *Dijkstra* é um algoritmo iterativo que, ao fim de n iterações, consegue descobrir os caminhos de custo mínimo de um encaminhador para os possíveis destinos.
- Os caminhos mais curtos são incluídos na tabela de encaminhamento, com os campos destino, custo e endereço do próximo encaminhador.

Tratando-se de um protocolo de encaminhamento dinâmico, por defeito, a cada 30 minutos o encaminhador volta a propagar pela rede o estado das suas ligações, e caso ocorra alguma alteração dentro dos 30 minutos, os encaminhadores recalculam o caminho para cada destino. No caso de domínios *OSPF* baseados em mais que uma área, estas anunciam entre si um sumário das rotas.

Com as tabelas de encaminhamento construídas e periodicamente atualizadas, os encaminhadores encontram-se em condições de encaminhar os pacotes que chegam pelos dispositivos de rede para o encaminhador seguinte, de acordo com as entradas nas tabela.

3.2 Encaminhamento com QoS

No modelo de melhor esforço, atualmente em vigor na *Internet*, os encaminhadores trocam mensagens entre si com informação relativa às suas ligações, especificamente com o estado e o "custo" das mesmas. Este "custo" é normalmente obtido a partir de uma métrica única, que varia de acordo com o protocolo de encaminhamento, podendo ser baseada apenas na largura de banda dísponível. Com a informação anunciada são calculados os caminhos mais curtos, relativos à métrica utilizada, podendo no entanto existir outros caminhos, mas que não são utilizados.

O objetivo do encaminhamento com Qualidade de Serviço é identificar caminhos mais curtos a partir de métricas baseadas nos requisitos de *QoS* das aplicações que geram o tráfego que circula na rede. Os pacotes de dados provenientes das aplicações com requisitos de *QoS* devem, de alguma forma, ser encaminhados por caminhos que satisfaçam os requisitos, quer seja efetuada reserva de recursos por fluxo, ou

instalando nas tabelas de encaminhamento diferentes caminhos por classe de serviço.

Podem ser utilizadas várias métricas simples para estabelecer caminhos a partir dos requisitos de *QoS* das aplicações, desde o número de saltos, a largura de banda disponível, o atraso, a percentagem de pacotes perdidos, etc.

A determinação do caminho mais curto quando é utilizada apenas uma métrica, pode ser resolvida utilizando adaptações dos protocolos de caminho mais curto; no entanto, em alguns casos não basta utilizar apenas uma métrica, mas sim uma combinação de métricas. Por exemplo, encontrar um caminho ótimo que satisfaça os requisitos de largura de banda mínima com um atraso mínimo. Para qualquer uma das situações, a mudança de paradigma obriga a considerar novos fatores, como por exemplo a natureza das métricas. As métricas podem ser classificadas em 3 categorias:

- Métricas Côncavas - O custo total de um percurso é igual ao *mínimo/máximo* custo observado. Por exemplo: a largura de banda disponível para um percurso fim a fim é igual à mínima largura de banda disponível observada ao analisar todas as ligações que compõem o percurso.
- Métricas Aditivas - O custo total de uma rota é o somatório dos custos de cada ligação que a compõe. Um exemplo de métrica aditiva é o atraso fim a fim, em que o atraso total do caminho percorrido é igual à soma dos atrasos em cada ligação do percurso.
- Métricas Multiplicativas - O custo total de uma rota é o produtório dos custos de cada ligação que a compõe. A percentagem de pacotes perdidos é um exemplo de uma métrica multiplicativa.

Vários algoritmos de encaminhamento com *QoS* têm sido propostos ao longo dos últimos anos, cada um deles com abordagens distintas, no sentido de resolver as diversas problemáticas associadas a este tipo de encaminhamento.

3.3 Algoritmos de encaminhamento com QoS

Nesta secção serão apresentados três algoritmos de encaminhamento com diferenciação de tráfego, desenvolvidos por grupos de investigação da área de encaminhamento de tráfego com *QoS*. O primeiro algoritmo é fruto de um estudo feito relativamente à partilha de recursos da rede por múltiplas classes de tráfego. O segundo algoritmo apresenta uma proposta de encaminhamento com *QoS* por classes

de serviço, com o objetivo de minimizar o impacto das classes de alta prioridade sobre o desempenho das classes de baixa prioridade. Por último, surge também uma proposta de encaminhamento com *QoS*, com o objetivo de otimizar o desempenho global sem afetar as classes individualmente.

3.3.1 Multi-Class QoS Routing Algorithm

Na tentativa de mudar a tendência da maioria dos estudos, na área de encaminhamento com *QoS*, que apenas se focam em algoritmos de encaminhamento que optimizam o débito da rede para classes de serviço individuais, foi proposto em [24] um algoritmo de encaminhamento dinâmico, que permite a partilha de recursos da rede entre múltiplas classes de serviço.

O algoritmo proposto é baseado na *largura de banda residual virtual*, obtida a partir da *largura de banda residual*, considerando o estado de congestão do tráfego de baixa prioridade na rede. O estudo mostra que a utilização da *largura de banda residual virtual*, na função de custo de ligação para classes com *QoS*, melhora o desempenho da rede na medida em que classes com *QoS* não prejudicam o desempenho das classes de baixa prioridade.

As classes de *QoS* utilizam a nova função de custo, baseada na *largura de banda residual virtual*, o que desencoraja este tipo de tráfego de utilizar ligações que se encontrem muito carregadas com tráfego *Best Effort*, se existirem caminhos alternativos.

Encaminhamento e partilha de recursos Multi-Classes

Numa rede onde circula tráfego de ambas as classes, *Best Effort* e *QoS* (esta engloba todas as classes prioritárias), a quantidade de recursos disponíveis para *Best Effort* vai depender da taxa de utilização das ligações pela classe de maior prioridade, classe *QoS*. Considerando que o tráfego *Best Effort* é, na maioria dos cenários, a classe dominante, otimizar o débito da classe prioritária com um algoritmo convencional, sem considerar o desempenho do *Best Effort*, pode provocar um aumento da congestão nas ligações. Este estudo focou-se em dois pontos chave:

- Admitir o mesmo número de sessões *QoS* que uma rede dedicada a este tipo de tráfego;
- Melhorar o desempenho do tráfego *Best Effort*, otimizando o estabelecimento de sessões *QoS*.

O objetivo passa por direcionar o tráfego *QoS* pelas ligações pouco carregadas de tráfego *BE*. A partir da *largura de banda residual* de uma ligação obtém-se a *largura de banda residual virtual*, que é inferior à real quando a ligação está muito carregada de *BE*, e superior no caso contrário.

O estado de congestionamento da ligação é determinado neste algoritmo pelo método *"max-min fair share rates"*, que funciona como uma espécie de barómetro do congestionamento da rede. São estabelecidos patamares de congestão, a partir dos quais se determina um valor para somar à *largura de banda residual*, denominado *credit BW*, obtendo-se a *largura de banda residual virtual*.

$$\text{virtual residual BW} = \text{residual BW} + \text{credit BW}$$

Sempre que a ligação estiver pouco congestionada é somado um crédito positivo, o que torna a ligação mais atrativa ao tráfego *QoS*. No caso da ligação estar muito congestionada é somado um crédito negativo, tornando a ligação pouco atrativa. Quando a ligação estiver no patamar intermédio de congestão não é somado qualquer valor, tornando as larguras de banda real e virtual iguais.

Ambiente de Simulação

Para proceder à avaliação do algoritmo foi usado um simulador baseado em eventos, que modela atividades em dois níveis. Ao nível de sessão é feita a seleção de rotas, controlo de admissão e reserva de recursos e, ao nível do pacote é configurada a conexão e distribuída a informação de encaminhamento.

São usadas duas topologias, uma topologia *MCI* (Figura 3.1) e uma *cluster* baseada em *switch* (Figura 3.2), e a carga de tráfego consiste em duas classes, *QoS* e *BE*, que serão uniformemente ou não uniformemente distribuídas na rede.

Além de informação da topologia, o estado de ligação vai conter a *largura de banda residual virtual* e o *max-min fair share rates*, informação propagada pelos nós, numa versão de *flooding* (inundação da rede), a cada 30 segundos.

A maior parte da avaliação centra-se na comparação entre o desempenho da abordagem proposta neste estudo, com abordagens em que o encaminhamento de tráfego das classes é efetuado individualmente.

O objetivo passa por determinar em que situações é desejável seguir a abordagem de partilha dinâmica de recursos *inter-classes*, e avaliar a capacidade de atingir o fim para que foi proposta, sem afetar o desempenho da rede.

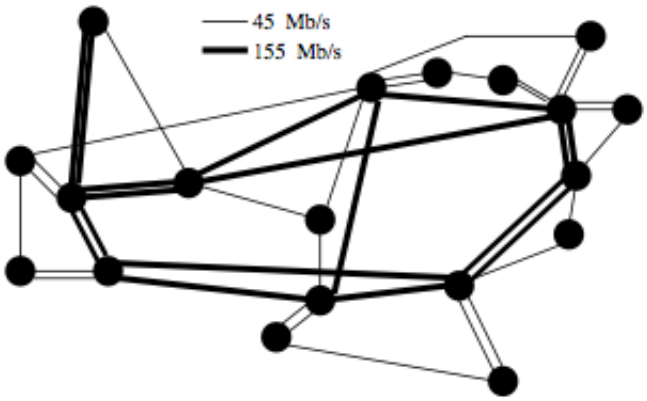


Figura 3.1: Topologia MCI

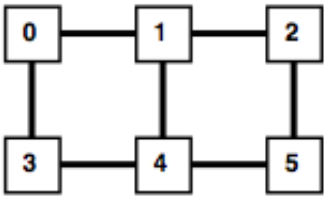


Figura 3.2: Topologia Cluster baseada em Switch

Avaliação do algoritmo

Ao analisar os resultados obtidos em [24], verificamos que, no caso da topologia *MCI* onde o tráfego é igualmente dividido entre *BE* e *QoS*, o melhor desempenho do algoritmo é alcançado quando a carga é não uniforme. Isto pelo facto de que o objetivo do algoritmo é orientar o tráfego *QoS* pelas ligações menos congestionadas com tráfego *BE* e no caso da distribuição uniforme, não existem caminhos especialmente congestionados. No caso da carga ser irregular, a melhoria alcançada com o algoritmo é já significativa.

Na topologia *cluster* de seis nós, onde o tráfego é igualmente dividido, tal como na topologia *MCI*, são apresentados quatro cenários de cargas uniformemente ou

não uniformemente distribuídas. Ao analisar os gráficos dos resultados em [24], verificamos que em nenhum deles o fluxo de *QoS* é bloqueado, mas no caso em que o tráfego *QoS* é não uniformemente distribuído e o tráfego *BE* é uniforme, existe uma degradação no desempenho do algoritmo baseado em *largura de banda residual virtual*. A razão para esta degradação é que com a utilização deste mecanismo, muitas vezes as sessões *QoS* usam caminhos ligeiramente mais longos, acabando por privar as sessões *BE* desses recursos.
Nos restantes cenários, no caso de carga irregular, verificamos também nesta topologia, uma melhoria significativa.

3.3.2 A QoS Based Routing Algorithm for Multi-Class Optimization in DiffServ Networks

No ambiente atual, com cada vez mais tráfego, e mais diversificado, a ser transmitido pelas redes IP, que apenas oferecem um serviço de melhor esforço, surge o modelo *DiffServ*, para suporte de diversas classes de tráfego com prioridades distintas.

O estudo apresentado em [25] propõe um esquema de encaminhamento com *QoS* por classes de serviço, designado *PERD*(Per-Classe Dissemination), que assenta no modelo *DiffServ*, numa tentativa de minimizar o impacto das classes de alta prioridade sobre o desempenho das classes de mais baixa prioridade.

O objetivo do sistema proposto por estes académicos, é permitir a otimização do encaminhamento, aplicando diferentes algoritmos na seleção de caminhos para as diferentes classes de serviço. Com o algoritmo proposto, a distribuição do tráfego pela rede depende do tráfego oferecido por cada classe de serviço e também da perceção do estado da rede de cada classe.

PERD

O *PERD* é caracterizado por permitir que diferentes classes tenham diferentes requisitos de *QoS*, é adequado para o modelo *DiffServ* com esquemas de encaminhamento multi-classe e é passível de ser implementado de forma centralizada ou distribuída. Neste algoritmo são calculadas as melhores rotas para cada classe de serviço, dependendo naturalmente dos requisitos de *QoS* das mesmas. Para descrever o funcionamento do *PERD* são utilizadas duas classes, *BE* e *EF* (Figura 3.3), sendo que o algoritmo pode ser aplicado a um número ilimitado de classes.

A escolha dos algoritmos de encaminhamento e estado de ligação não é especificada neste esquema, podendo os mesmos serem escolhidos independentemente.

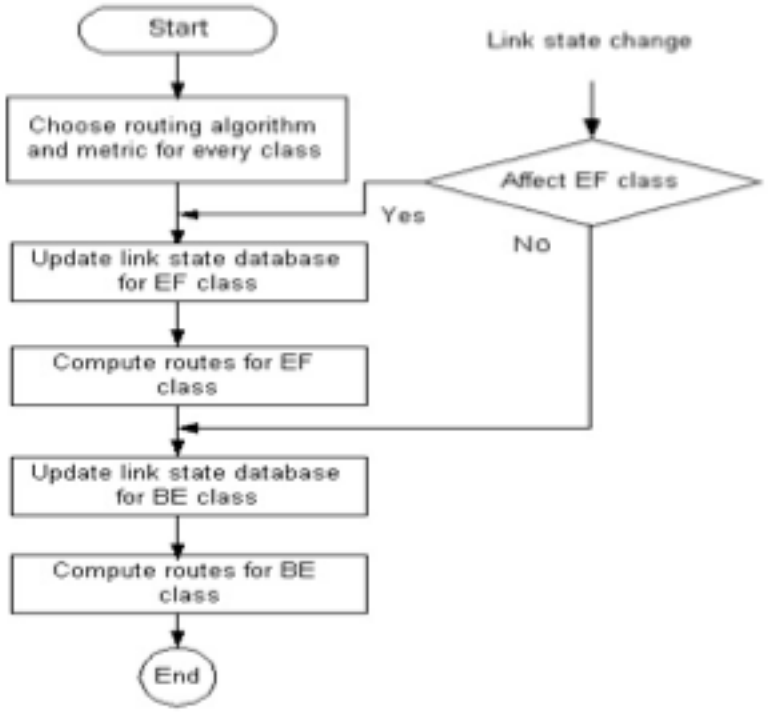


Figura 3.3: Fluxograma PERD [25]

De acordo com os requisitos de *QoS* e de otimização são definidos inicialmente os algoritmos de encaminhamento e as métricas para cada classe. De seguida, começando na classe de maior prioridade, são calculadas as rotas de acordo com o estado de ligação percecionado pela classe; este processo repete-se até todas as classes terem calculado as melhores rotas e sempre que ocorram atualizações no estado de ligação.

O algoritmo pode ser implementado como uma extensão do *OSPF*, em que cada nó da rede contém uma tabela de encaminhamento por classe de serviço, construída a partir do estado de ligação de cada classe.

Ambiente de Simulação

De forma a avaliar o desempenho do algoritmo *PERD* para cada classe, em termos de rendimento, atraso e custo em tempo de processamento, nas simulações é utilizada uma rede *MPLS DiffServ* onde circulam três classes de tráfego, *BE, AF e EF*.

O *PERD* é analisado comparativamente ao algoritmo convencional da *Internet*,

o de caminho mais curto (SP - Shortest Path) e ao algoritmo de maior largura de banda (WB - Widest Bandwith). Sendo que o *PERD* permite a utilização de diferentes algoritmos para diferentes classes, será utilizado o *WB* para as classes *AF e EF* e o algoritmo *SP* para a classe *BE*.

Nas simulações, os recursos da rede são previamente atribuídos às classes, respetivamente 20% da capacidade da ligação para *EF*, 30% para *AF* e o restante para *BE*, sendo que esta classe agrega a maioria do tráfego que circula na rede e pretende-se que esta nunca seja prejudicada pelas classes de prioridade superior.

O simulador utilizado neste estudo foi o *QRS* (QoS Routing Simulator), onde o tráfego flui dos nós fonte para os nós destino, com um aumento progressivo da carga de 10% de cada vez, até atingir a percentagem máxima de carga suportada pela rede.

Avaliação do Algoritmo PERD

Ao analisar os resultados obtidos nas simulações [25], verificamos que atingimos um melhor desempenho com a utilização do algoritmo proposto (PERD), sendo que são otimizadas todas as classes de serviço com base na informação do estado de ligação de cada uma.

Nas simulações o *PERD* mostrou-se como o algoritmo que mais rendimento confere a cada classe, principalmente às mais prioritárias; no entanto, relativamente ao atraso fim a fim, o algoritmo que resulta numa melhor performance é o SP, pelo facto de selecionar os caminhos mais curtos, seguido do *PERD* que é superior ao *WB* para todas as classes. Os custos de processamento registados são semelhantes no *PERD* e no *WB*.

Este trabalho explorou o facto de os algoritmos tradicionais não serem capazes de alcançar a otimização de diversas classes de tráfego em simultâneo. Desta forma foi proposto o algoritmo *PERD* para resolver este problema de uma forma heurística.

3.3.3 Multiclass QoS Routing Strategies Based on the Network State

Com o estudo [26] surge mais uma perspetiva de encaminhamento com qualidade de serviço, para múltiplas classes de tráfego, baseado, tal como em [24], no conceito de *largura de banda residual virtual* associada ao estado da rede.

O principal objetivo passa por melhorar o desempenho da classe de serviço de mais baixa prioridade, *BE*, sem prejudicar o desempenho das classes de *QoS*, de maior prioridade. O conceito abordado prende-se com o facto de que o algoritmo de

encaminhamento de tráfego individual não muda, a única mudança é a definição do custo de ligação.

Em estudos anteriores ([27] e [24]), verificamos que este tipo de abordagem é eficaz, no entanto em todas as situações regista-se uma pequena degradação do desempenho do tráfego *BE*. Neste algoritmo, como será explicado de seguida, é possível otimizar o rendimento global, sem afetar negativamente o desempenho das classes individualmente, fazendo uma utilização eficiente dos recursos da rede.

Algoritmo

Tal como demonstrado nos outros estudos abordados no estado da arte [24],[25], os recursos disponíveis para o tráfego *BE* dependem do encaminhamento das classes de maior prioridade, as classes *QoS*. Com este algoritmo pretende-se melhorar o desempenho do tráfego de baixa prioridade, sem afetar as classes de *QoS*.

O algoritmo baseia-se na utilização da *largura de banda residual virtual* como função de custo, para as classes de *QoS*, mas não de forma linear. A utilização da *largura de banda residual virtual* vai depender do estado da rede, da carga e da distribuição do tráfego na mesma. Aqui reside o factor chave do algoritmo.

Vão ser estabelecidos dois patamares: um representativo da utilização média da ligação pelo tráfego das classes *QoS* e um outro patamar para o coeficiente de variação de tráfego *BE*. O primeiro patamar estabelecido é usado para diferenciar o nível de carga de tráfego *QoS* nas ligações, ao passo que o segundo patamar indica o tipo de distribuição do tráfego *BE*, mais ou menos uniforme. Desta forma, dependendo destes dois limitadores, o tráfego *QoS* vai utilizar na função de custo, a*largura de banda residual virtual*, ou a *largura de banda residual real*, a fim de evitar ligações muito congestionadas com tráfego *BE*.

Nesta abordagem, a *largura de banda residual virtual* será substancialmente superior à real, em ligações pouco congestionadas, no sentido de tornar estas ligações atrativas ao tráfego *QoS*, ao passo que em ligações muito congestionadas com tráfego *BE*, a *largura de banda residual virtual* será aproximadamente igual à real.

A nova função de custo para classes de *QoS* depende da utilização da ligação pelo tráfego *BE*, e da utilização total da ligação:

$$\textbf{New\_Virtual\_Residual\_Bandwidth} = (1+\alpha) \times \textbf{Real\_Residual\_Bandwidth} \qquad (0 \leq \alpha \leq 1);$$

$$\alpha = (1-\text{total\_link\_utilization}) \times (1-\text{Best\_Effort\_link\_utilization}).$$

Ambiente de simulação

Para testar o algoritmo foi utilizada uma topologia em malha (Figura 3.4), com nove encaminhadores, na qual todas as ligações têm igual capacidade, *100 Mbps*. Foram utilizadas duas classes de tráfego, *BE* e *QoS*, sendo que o tráfego *QoS* chega com uma distribuição exponencial, tem duração média de *180 segundos* e solicita uma largura de banda de *1,5 Mbps*. O tráfego *BE* varia entre *1 Mbps* e *80 Mbps*.

Na simulação serão analisados dois cenários, um em que o tráfego *BE* percorre apenas 4 ligações da topologia e outro onde o tráfego menos prioritário percorre todas as ligações disponíveis, ou seja, é uniformemente distribuído.

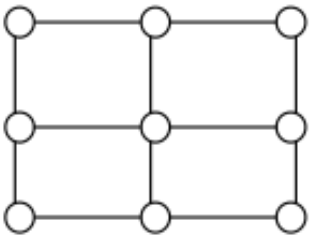


Figura 3.4: Topologia em Malha

Avaliação do algoritmo

Para determinar o impacto destes novos fatores (utilização média da ligação e variação da distribuição de tráfego *BE*), sobre o desempenho de toda a rede, foram alvo de comparação durante a simulação a nova abordagem com a abordagem sugerida pelos mesmo académicos em [27]. Esta última corresponde, em traços gerais, a disponibilizar ao tráfego *QoS* a largura de banda não utilizada pelo tráfego *BE*.

Os resultados obtidos em [26] mostram que o desempenho do tráfego *BE*, nesta nova abordagem, é melhorado, em comparação com o algoritmo convencional.

Quando a carga de tráfego *QoS* é leve, verificamos uma melhoria muito significativa no primeiro cenário, em o tráfego *BE* apenas circula em quatro ligações da topologia. No caso em que o tráfego *BE* é uniformemente distribuído pela rede, não ocorre nenhuma melhoria relevante, não havendo, no entanto, nenhuma degradação no desempenho do tráfego *QoS*.

À medida que aumentamos o primeiro patamar, limite para utilização média da ligação pelo tráfego *QoS*, o desempenho do tráfego *BE* melhora, sendo que o desempenho do tráfego *QoS* se mantém idêntico.

Relativamente ao segundo limitador, distribuição de tráfego *Best-Effort*, as simulações mostram que nem sempre a utilização desta nova abordagem é ideal visto

que, em situações em que a distribuição do tráfego *Best-Effort* pela rede é muito uniforme, verificamos um aumento na taxa de bloqueio de tráfego *QoS*, sendo minimizado este fenómeno quando é utilizado o algoritmo convencional.

O algoritmo proposto neste estudo, mostrou-se também muito eficiente quando aplicado a topologias mais extensas em número de ligações.

Capítulo 4

Encaminhamento com Classes de Serviço: Uma Proposta

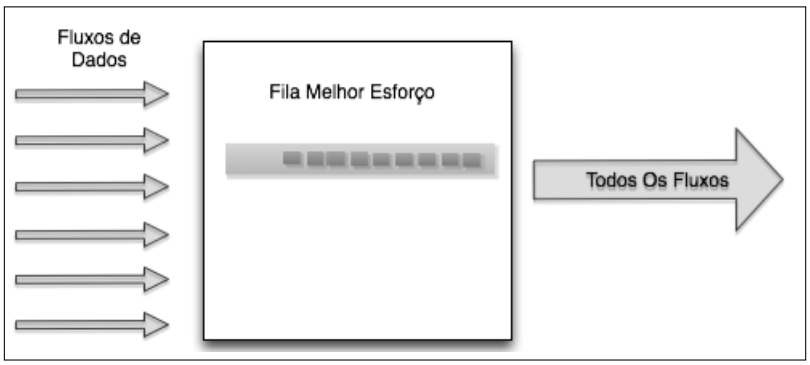
O encaminhamento por classes de serviço tem sido foco de estudo em diversos núcleos de investigação da área das redes de computadores, como foi descrito no capítulo 3, no entanto não foi ainda alcançada uma solução plena que possa ser implementada a uma escala global, no encaminhamento inter-domínio.

Neste capítulo é descrita a estratégia proposta para introduzir qualidade de serviço na camada de rede do protocolo *IP*, especificamente ao nível do encaminhamento. A estratégia de encaminhamento de tráfego por classes de serviço assenta no modelo de diferenciação de tráfego proposto pelo *IETF*, o *DiffServ*[14] e no protocolo de encaminhamento global centralizado *OSPF*.

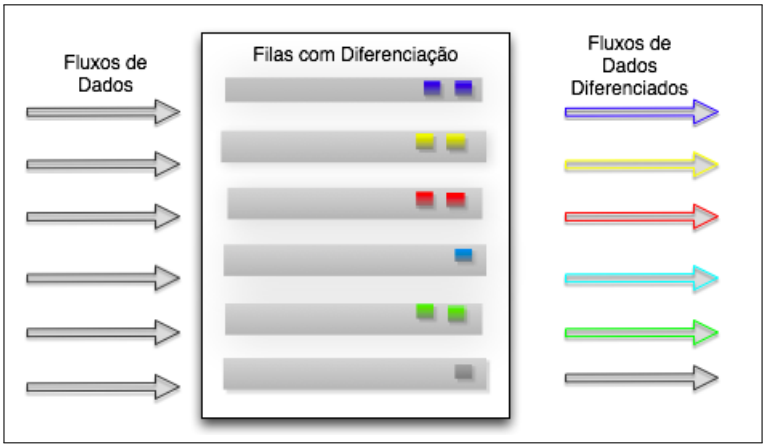
4.1 Diferenciação do tráfego

A estratégia de encaminhamento por classes de serviço pressupõe, numa primeira fase, que seja efetuada a diferenciação do tráfego proveniente das aplicações. Este processo envolve vários intervenientes, desde o cliente das aplicações geradoras de tráfego, o provedor de serviços da internet *ISP*, até aos encaminhadores no domínio de diferenciação da rede. O Cliente acorda com o *ISP* um nível de serviço para determinado fluxo, com mais ou menos garantias dependendo dos requisitos da aplicação. Todos os pacotes pertencentes ao fluxo são depois marcados com a respetiva classe de serviço, para que posteriormente os encaminhadores acondicionem todo esse tráfego de acordo com os parâmetros acordados. A classe que mais garantias de *QoS* oferece é a *EF*, seguida das classes *AF*(*AF1, AF2, AF3 e AF4*) e por último a classe *BE*.

No cenário atual de melhor esforço, os dispositivos de rede dos encaminhadores incluem apenas uma fila de espera para onde são redirecionados os pacotes de dados, enquanto aguardam para ser novamente enviados pelos canais de comunicação. Na Figura 4.1 é representado o tratamento dado aos fluxos de dados nas duas abordagens, sem diferenciação e com diferenciação nas filas dos encaminhadores.



(a) Modelo de melhor esforço



(b) Modelo DiffServ

Figura 4.1: Filas sem diferenciação *vs* filas com diferenciação

Como podemos observar na Figura 4.1(b), no cenário com diferenciação adotado é necessário que exista uma fila por cada uma das classes de serviço. A existência de uma fila por cada classe de serviço vai permitir implementar diferentes mecanismos de escalonamento e gestão nas filas mediante os requisitos das classes, além de auxiliar, como vai ser explicado de seguida, no processo de monitorização do desempenho da rede. Com este tipo de abordagem, os fluxos recebem na rede um tratamento diferenciado, ao contrário da abordagem tradicional, onde os fluxos são tratados pela rede sem considerar requisitos de *QoS*.

O modelo *DiffServ* do *IETF*[14], descrito no capítulo do estado da arte, baseia o seu funcionamento nos pressupostos supracitados, daí ter sido escolhido como o

modelo para efetuar a diferenciação do tráfego nos encaminhadores da rede.

4.2 Monitorização do desempenho das classes

Considerando que os pacotes pertencentes a fluxos de tráfego com requisitos de *QoS* são previamente marcados com a classe de serviço, e depois nos encaminhadores redirecionados para a fila de comportamento associada à classe, nesta segunda etapa vai ser analisado o desempenho das classes, monitorizando as filas dos dispositivos de rede periodicamente. Deve ser efetuado um registo detalhado dos tempos de entrada e saída de todos os pacotes em cada fila dos encaminhadores, sendo esta a chave para o processo de monitorização. Com os valores registados é possível determinar o tempo que os pacotes ficam em espera na fila, se os pacotes são descartados, e o tamanho da fila em cada momento.

A monitorização é um processo periódico e iterativo que abrange todos os encaminhadores do domínio de diferenciação e todas as filas dentro de cada encaminhador, num intervalo definido por configuração. O objetivo é obter parâmetros de desempenho associados a cada fila, para determinar o nível de congestionamento da rede e, caso necessário, alterar o custo das ligações entre os encaminhadores.

Numa arquitetura sem diferenciação, o custo das ligações está por norma associado à largura de banda disponível entre os encaminhadores. Nesta estratégia o custo das ligações vai depender de parâmetros observados no processo de monitorização, e a mesma ligação pode ter custos associados distintos, um por cada classe de serviço.

4.2.1 Medidores de desempenho

Os parâmetros escolhidos para determinar o desempenho de cada classe depende mais uma vez dos requisitos da aplicação. Considerando os comportamentos normalizados, referidos no modelo *DiffServ* no capítulo 2.2.3, para o tráfego associado à classe *AF* o valor alvo de monitorização, nas filas para onde é escalonado, será a percentagem de pacotes perdidos.

Percentagem de perda de pacotes A percentagem de perda de pacotes representa, em termos percentuais, o número de pacotes que foram redirecionados para determinada fila dentro do encaminhador, mas que foram por algum motivo descartados, não sendo encaminhados pelo dispositivo de rede de saída.

O tráfego pertencente à classe de serviço *EF* é também sensível à perda de pacotes, mas é especialmente exigente em termos de atraso, por esse motivo o valor do atraso por fila será o alvo de monitorização.

Atraso médio O atraso na fila representa o tempo que o pacote esperou na fila antes de ser transmitido para a rede, sendo o atraso médio calculado a partir dos atrasos registados de todos os pacotes que entraram na fila.

O restante tráfego sem marcação é redirecionado para as filas *BE* e não será alvo de monitorização, mantendo-se inalterado o custo da ligação determinado na configuração.

Tanto a percentagem de pacotes perdidos como o atraso médio são obtidos a partir dos tempos de entrada e saída das filas num determinado intervalo de tempo de simulação, sendo esse intervalo menor ou maior, dependendo de pretendermos obter valores instantâneos ou históricos. Na estratégia proposta, no sentido de tornar o modelo mais sensível a alterações do estado das ligações, foi definido que seriam usados parâmetros de desempenho calculados com os valores observados em cada intervalo de monitorização.

Na Figura 4.2 podemos observar um diagrama temporal ilustrativo do processo de monitorização durante uma simulação, onde o período de monitorização está definido para cada 2 segundos e ocorre atualização dos custos apenas após a segunda monitorização.

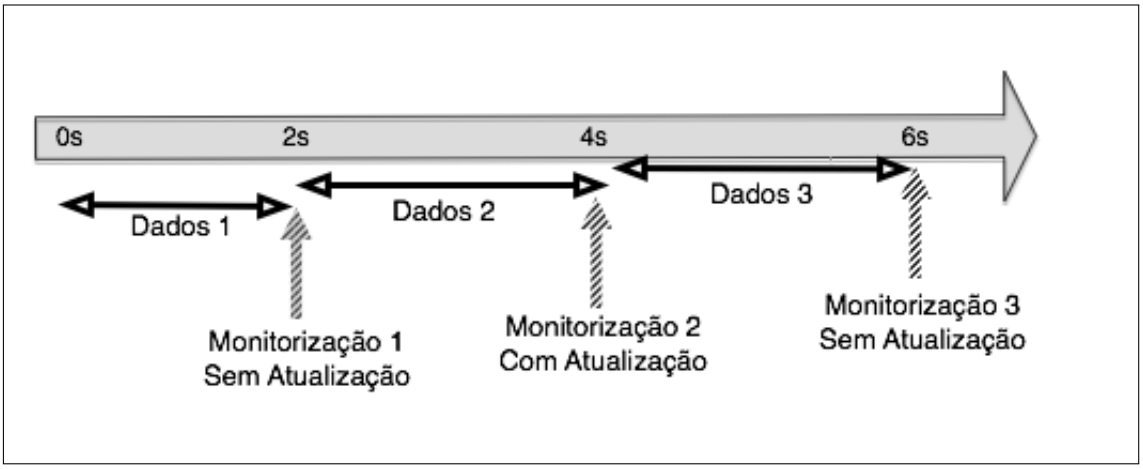


Figura 4.2: Processo de Monitorização

No instante $t = 0s$ são divulgados os valores médios da percentagem de pacotes perdidos e de atraso. No instante $t = 2s$ é efetuada a primeira monitorização, são

calculados os parâmetros de desempenho das filas a partir dos valores registados no intervalo *[0s, 2s]*, e não se verifica uma alteração significativa dos parâmetros relativamente aos previamente divulgados logo, não é necessário proceder a uma atualização dos custos e os novos valores não são divulgados.

Na segunda monitorização ao instante $t = 4$, são usados os valores registados no intervalo de tempo *[2s, 4s]*, verifica-se que ocorreu uma alteração significativa sendo desta forma necessário proceder à atualização de algum dos custos. Os novos valores são divulgados para serem usados como comparação na monitorização seguinte.

Na última monitorização apresentada na figura, instante $t = 6$, o cálculo dos parâmetros de desempenho é baseado nos valores registados no intervalo *[4s, 6s]* e, mais uma vez, não se verifica a necessidade de atualizar o custo das ligações, não sendo desta forma divulgados os novos valores dos parâmetros de desempenho.

É de salientar que na estratégia adotada este processo é independente de classe para classe, ou seja, o facto de ser detetada a necessidade de atualização de custos durante a monitorização a uma das filas, significa que são especificamente divulgados os novos parâmetros de desempenho relativos à fila em questão.

4.2.2 Atualização dos custos das ligações

Em cada fila dos encaminhadores é feita uma comparação entre os parâmetros monitorizados e os parâmetros divulgados na monitorização com atualização anterior e, se a diferença entre eles for muito significativa, então é necessário proceder a uma atualização do custo da ligação associada a essa fila, no sentido de a tornar mais ou menos atrativa. Na Figura 4.3 é apresentado, na forma de fluxograma, o processo de monitorização a uma fila de tráfego *AF*. O processo de monitorização às restantes filas das classes de *QoS* processa-se da mesma forma, apenas variando os parâmetros de desempenho alvo de análise.

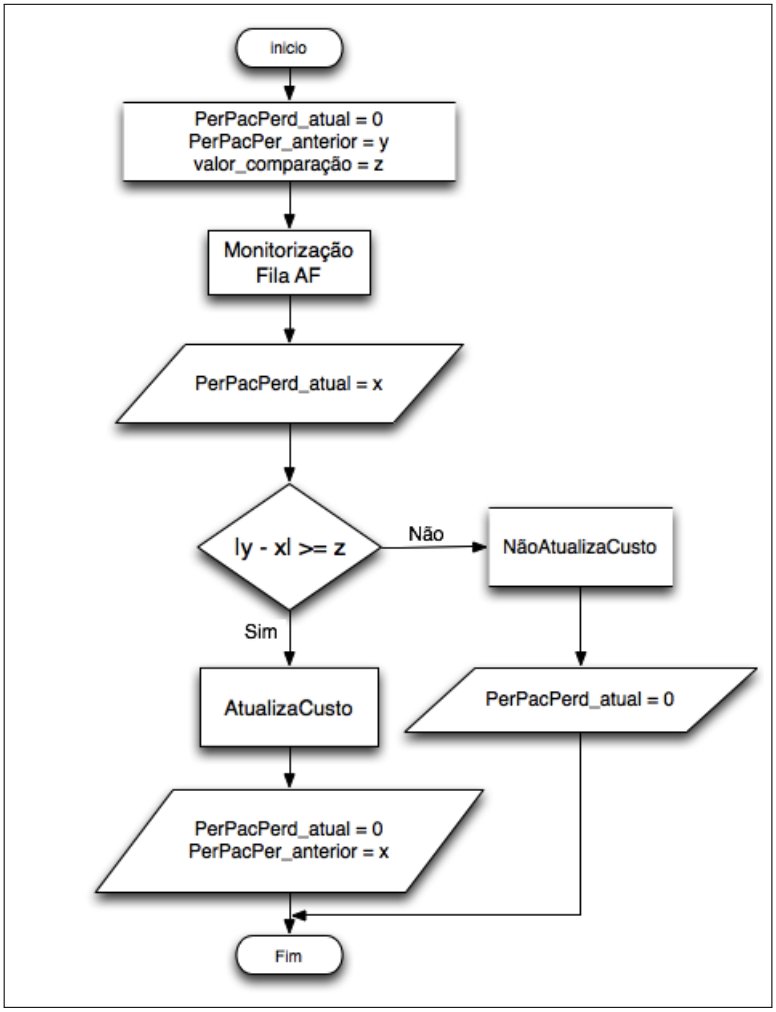


Figura 4.3: Fluxogramas da monitorização às filas AF

A diferença entre registos ser ou não significativa depende do valor usado como termo de comparação definido na configuração, representado no fluxograma por *"valor_comparacao"*. Quanto menor for, maior será a probabilidade de ser necessário atualizar o custo das ligações o que pode aumentar em demasia a sobrecarga introduzida pelas mensagens de controlo, e o grau de processamento nos encaminhadores. Por outro lado, um valor de comparação muito alto pode prejudicar o desempenho da estratégia, não sendo realizadas atualizações suficientes, aproximando-a do modelo de melhor esforço.

Na estratégia desenhada, o custo está diretamente relacionado com os parâmetros de desempenho de cada classe, analisados na monitorização. O custo de uma ligação para o tráfego *AF* é baseado na percentagem de pacotes perdidos, e quanto maior for a percentagem de pacotes perdidos maior será o custo da ligação e vice-versa. O custo de uma ligação para o tráfego *EF* é baseado no atraso médio da fila,

sendo o custo maior quanto maior for o atraso observado.

A título de exemplo, na Figura 4.4 é representada uma topologia muito simples com apenas 3 encaminhadores e dois canais de comunicação, onde podemos identificar os pacotes de dados das diferentes classe de serviço (representados na imagem com cores distintas), a percorrer o caminho desde a origem *10.0.0.1* até ao destino *10.0.0.3*. São também apresentadas duas Tabelas (4.1 e 4.2) onde constam os valores médios de percentagem de pacotes perdidos e atraso observados pelas classes *EF, AF e BE*, nas filas dos encaminhadores 1 e 2, respetivamente. Os parâmetros observados nas filas do encaminhador 3 não constam no exemplo porque este se trata do encaminhador destino, não sendo estes valores considerados no cálculo do custo total do caminho.

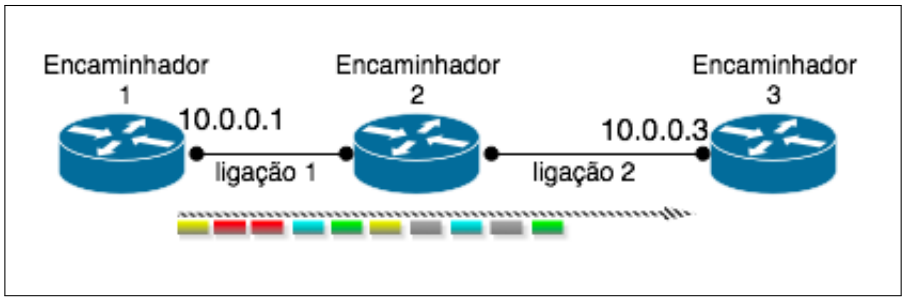


Figura 4.4: Exemplo de um cenário de encaminhamento

Encaminhador DiffServ 1			
	EF	BE	AF
Média Percentagem pacotes perdidos	10%	x	5%
Média do atraso	0,02	x	1,1

Tabela 4.1: Parâmetros de desempenho nas filas do Encaminhador 1

Encaminhador DiffServ 2			
	EF	BE	AF
Média Percentagem pacotes perdidos	9%	x	3%
Média do atraso	0,04	x	1,2

Tabela 4.2: Parâmetros de desempenho nas filas do Encaminhador 2

Com os valores apresentados nas Tabelas 4.1 e 4.2 é possível calcular o custo de cada ligação e posteriormente o custo total do caminho percorrido pelos pacotes

desde a origem até ao destino. O custo de cada ligação para cada uma das classes de serviço aqui representadas, cumprindo os pressupostos de funcionamento da estratégia são:

	Ligação 1	Ligação 2
EF	0,02	0,04
AF	0,05	0,03
BE	1	1

Tabela 4.3: Custo das ligações 1 e 2 para as diferentes classes de serviço

Ao analisar a tabela 4.3, verificamos que os custos percecionados pelas 3 classes são distintos. Para a classe de serviço *EF* cada ligação tem um custo igual ao valor de atraso médio calculado na monitorização da fila. Para a classe de serviço *AF* o custo da ligação é igual à média de pacotes perdidos. Para a classe de seviço *BE*, tal como foi já referido, o custo da ligação tem um valor fixo de 1, não sendo estas filas alvo de análise no processo de monitorização.

Considerando que a mesma ligação tem um custo associado por cada uma das classes de tráfego, na estratégia adotada a tabela de encaminhamento de cada encaminhador vai ter tantas entradas para um destino quantas as classes de tráfego. Ou seja, o melhor caminho percecionado por uma das classes de serviço não é necessariamente o melhor caminho percecionado por uma classe de serviço distinta. Continuando no exemplo apresentado na Figura 4.4, vamos agora proceder ao cálculo do custo do caminho fim a fim (Tabela 4.4).

	Custo fim a fim
EF	0,06
AF	0,08
BE	2

Tabela 4.4: Custo fim a fim por classe de serviço

Ao analisar a tabela verificamos que o custo do caminho é distinto para cada classe, no entanto, não significa que alguma classe seja beneficiada relativamente a outra no processo de encaminhamento. As classes têm uma visão independente umas das outras do estado da rede. Numa topologia típica *ISP*, com vários caminhos alternativos, o cálculo dos caminhos mais curtos de cada classe vai basear-se exclusivamente nos valores observados nas filas dessa classe.

A atualização dos custos das ligações é da responsabilidade do protocolo de encaminhamento, assim, no caso de ser determinado no processo de monitorização que é necessário alterar o custo de alguma das ligações dentro do domínio de diferenciação, é evocada a função do protocolo global centralizado *OSPF*, para que sejam recalculadas as tabelas de encaminhamento com os novos custos.

4.3 Alteração do processo de encaminhamento

No processo de encaminhamento tradicional, quando um pacote chega por um dispositivo de entrada do encaminhador, é analisado o seu cabeçalho *IP* para determinar qual o destinatário, e é de seguida selecionado na tabela de encaminhamento o melhor caminho, de forma a direcionar o pacote pelo dispositivo de saída correto. Na estratégia proposta, quando um pacote chega a um encaminhador dentro do domínio de diferenciação, pela análise do cabeçalho *IP*, além do destinatário, é determinada a classe de serviço a que o pacote pertence. Estes dois campos vão permitir selecionar na tabela de encaminhamento o caminho de custo mínimo para aquele destino, percecionado na classe de serviço específica. Na Figura 4.5 é apresentado um excerto de uma tabela de encaminhamento construída num contexto de simulação onde é aplicada a estratégia desenhada.

```
Node: 0 Time: 22s Ipv4ListRouting table
Priority: 0 Protocol: ns3::Ipv4GlobalRouting
Destination     Gateway         Genmask         Classe      Flags     Custo     Ref   Use Iface
10.0.26.1       10.0.3.2        255.255.255.255   AF1         UH        4         -     -   3
10.0.26.1       10.0.3.2        255.255.255.255   AF2         UH        4         -     -   3
10.0.26.1       10.0.3.2        255.255.255.255   AF3         UH        4         -     -   3
10.0.30.1       10.0.1.2        255.255.255.255   AF4         UH        12        -     -   1
10.0.26.1       10.0.1.2        255.255.255.255   EF          UH        7         -     -   1
10.0.26.1       10.0.3.2        255.255.255.255   BE          UH        4         -     -   3
```

Figura 4.5: Tabela de Encaminhamento

A Figura 4.5 apresenta um excerto da tabela de encaminhamento do *"node 0"*, no instante $t = 22$, com os caminhos para o destino *10.0.26.1*. Além de outra informação de encaminhamento podemos identificar o custo do caminho mais curto associado a cada classe de serviço no campo *"Custo"*. Segundo a tabela de encaminhamento, o melhor interface de saída para encaminhar o tráfego pertencente às classes *AF1, AF2, AF3, e BE* é o *10.0.3.2*. Paras as classes de serviço *AF4 e EF* será o interface *10.0.1.2*.

Os custos apresentados neste excerto da tabela de encaminhamento do *"node 0"* foram obtidos depois de aplicado o algoritmo de caminho mais curto do protocolo de encaminhamento *OSPF*, significando que estes caminhos são os que melhor satisfazem os requisitos de cada uma das classes de serviço independentemente.

4.4 Discussão

No presente capítulo é descrita a estratégia proposta para encaminhamento com classes de serviço. São especificados os mecanismos e protocolos adotados e os requisitos que devem cumprir no sentido de atingir os objetivos propostos.

A estratégia divide-se em 3 áreas fundamentais: a diferenciação de tráfego ao nível das filas dos encaminhadores; a monitorização das filas dos encaminhadores para atribuição de custos às ligações; e a alteração do processo de encaminhamento.

A diferenciação de tráfego nas filas vai permitir que os pacotes de dados recebam um tratamento adequado aos requisitos da classe de serviço a que pertencem. A este nível pretende-se que a tráfego com requisitos de *QoS* mais exigentes seja concedida maior prioridade do que ao tráfego com requisitos de *QoS* menos exigentes. São definidas 6 classes de serviço (*AF1, AF2, AF3, AF4, EF e BE*), logo seis filas distintas, em termos de precedência e nível de descarte, em cada dispositivo de rede dos encaminhadores. Além da questão da prioridade entre classes, este processo de diferenciação nas filas vai auxiliar na etapa seguinte, a monitorização do desempenho das filas de acordo com as classes de serviço.

A monitorização é o processo que confere o dinamismo à estratégia. A partir dos valores obtidos neste processo são calculados os custos das ligações entre os nós para as diferentes classes de serviço. Num período definido na configuração, iterativamente, cada uma das filas, em cada encaminhador do domínio de diferenciação, será alvo de monitorização. Com os tempos de entrada e saída dos pacotes das filas são calculados os parâmetros de desempenho associados a cada classe: Percentagem de pacotes perdidos para as classes *AF* e Atraso médio na fila para a classe *EF*. Durante a monitorização os valores calculados são comparados com valores obtidos na monitorização anterior. Se os valores forem significativamente distinto, são divulgados novos valores para que sejam atualizados os custos das ligações.

Os novos custos obtidos são propagados pelos encaminhadores da rede na forma de *LSAs*, pelo protocolo *OSPF*, e são calculados os caminhos mais curtos para cada uma das classes de serviço pelo protocolo de caminho mais curto. Os caminhos ob-

tidos são inseridos na tabela de encaminhamento que passa a ter um caminho para cada destino, por cada uma das 6 classes de serviço. A inclusão de caminho por classe de serviço altera significativamente o processo de encaminhamento. Quando um pacote de dados chega ao encaminhador por um dos seus interfaces de entrada, é analisado o cabeçalho do pacote para identificar o destinatário e o campos *ToS*, que contém a marcação referente à classe de serviço a que o pacote pertence. Com estes dois parâmetros é escolhido um caminho pela tabela de encaminhamento, para o destino pretendido de acordo com a classe de serviço específica.

Com esta estratégia pretende-se melhorar a gestão dos recursos da rede, conferindo um desempenho às classes de serviço, de acordo com os seus requisitos de funcionamento. Com a inclusão nas tabelas de encaminhamento de rotas por classe, o tráfego deixa de ser encaminhado na rede considerando apenas uma métrica simples, passando rota a ser calculada de acordo com métricas relacionadas com os requisitos da classe. Cada classe de serviço é tratada nesta estratégia de forma absolutamente independente, nos encaminhadores da rede é dado um tratamento diferenciado aos pacotes de cada classe e as rotas são calculadas especificamente para cada uma delas.

Capítulo 5

Implementação

Neste capítulo é feita a descrição detalhada da implementação da estratégia proposta bem como dos mecanismos de suporte utilizados.

Dada a complexidade associada a uma implementação num contexto real, foi utilizado o ambiente de simulação, especificamente o simulador *NS-3*, de forma a verificar o comportamento da estratégia num ambiente controlado.

5.1 Ambiente de simulação - NS-3

Os ambientes de simulação têm cada vez mais um papel decisivo no projeto de redes de computadores, pois permitem avaliar sistemas muito complexos, a um custo reduzido. Os simuladores permitem, no âmbito académico ou industrial, analisar e melhorar o desempenho dos protocolos de comunicação em diversos cenários de execução. É fundamental que as plataformas sejam abertas e suficientemente escaláveis para incluir a diversidade de elementos que vão sendo desenvolvidos pelas entidades de produção e investigação no setor.

Os simuladores de redes não conseguem replicar exatamente um sistema real, no entanto conseguem uma aproximação suficiente para que, depois de modelado o sistema, fazendo variar as características do mesmo, seja possível fazer uma análise dos resultados. Os resultados são tipicamente expressos de acordo com parâmetros de desempenho, como atraso fim a fim, largura de banda, pacotes descartados, etc.

Existem alguns simuladores de redes disponíveis no mercado, uns comerciais, que não disponibilizam o código fonte, como o *OPNET*[28] ou o *QualNet*[29], outros de código aberto, como o *NS-2* [30], o *NS-3*[31], o *OMNeT++*[32], etc. Cada simulador tem características próprias de funcionamento, sendo que a escolha do simulador depende do modelo a representar.

O *NS-2* é o simulador mais famoso no seio da comunidade académica, pelo facto de se tratar de um simulador com código fonte aberto e pela extensa biblioteca de componentes. No entanto, com base em experiências bem e mal sucedidas no *NS-2*, surgiu o *NS-3* que se encontra em fase de desenvolvimento e testes. O simulador *NS-3*, por todo o peso académico que tem, pela sua estrutura modular extremamente bem documentada, e pelo suporte para os mecanismos essenciais à implementação da estratégia proposta, foi o escolhido para representar o modelo de encaminhamento com diferenciação.

Network Simulator 3

O *Network Simulator 3* (*NS-3*), é um simulador de redes, baseado em eventos discretos [31]. O seu desenvolvimento teve inicio em 2006, com o objetivo de apoiar pesquisas, essencialmente no âmbito académico, na área das tecnologias da *Internet*. Todo o design assenta na linguagem de programação C++, e opcionalmente podem ser usadas *Scripts* Python, no apoio à simulação. Sendo um projeto relativamente recente, existem vários grupos de trabalho para a manutenção e desenvolvimento de novos modelos para o simulador.

O simulador está dividido num número relativamente alargado de módulos, que implementam os modelos que representam os elementos de rede e protocolos do mundo real.

Depois de escolhido o simulador passou-se à instalação e configuração do mesmo. Para esse efeito, em [33] é disponibilizado um guia de instalação detalhado, que faz uma pequena introdução acerca das plataformas que suportam o o simulador, bem como dos requisitos mínimos para correr as simulações. Este trabalho foi totalmente desenvolvido em sistema operativo *Mac OS 10.7*[34] com compilador *GNU gcc* versão 4.2 [35]. A versão do simulador instalada foi a *ns-3-allinone* [36], correspondente, na data de instalação, à versão estável *ns-3.16*.

Depois de efetuado o *download* do simulador, pela linha de comandos (Terminal Mac OS) foi construído o repositório local com a execução do comando *"./build"* e por último efetuadas as configurações com a ferramenta de configuração e instalação *Waf*. Para visualizar todos os módulos construídos e disponíveis para ser utilizados no contexto de simulação é evocado o comando *"./waf"* na linha de comandos. Na Figura 5.1 podemos observar o resultado da sua evocação.

```
MacBook-de-susana:ns-3-dev susanasousa$ ./waf
Waf: Entering directory `/Users/susanasousa/ns-3-allinone/ns-3-dev/build'
Waf: Leaving directory `/Users/susanasousa/ns-3-allinone/ns-3-dev/build'
'build' finished successfully (9.829s)

Modules built:
antenna                  aodv                     applications
bridge                   buildings                config-store
core                     csma                     csma-layout
diff-serv-queue (no Python) dsdv                    dsr
energy                   fd-net-device            flow-monitor
internet                 lte                      mesh
mobility                 mpi                      netanim (no Python)
network                  nix-vector-routing       olsr
point-to-point           point-to-point-layout    propagation
spectrum                 stats                    test (no Python)
tools                    topology-read            uan
virtual-net-device       wifi                     wimax

Modules not built (see ns-3 tutorial for explanation):
brite                    click                    emu
openflow                 tap-bridge               visualizer

MacBook-de-susana:ns-3-dev susanasousa$
```

Figura 5.1: Módulos NS-3

Para utilizar os módulos no âmbito das simulações são desenvolvidas *Scripts* em C++ ou Python, onde são configurados os cenários de simulação, tipicamente topologias de rede com atributos específicos. O escalonador do *NS-3* vai conter uma lista de eventos a serem executados num determinado tempo de simulação, sendo que o primeiro evento é executado quando é evocado o método *Simulation::Run*, e a simulação termina num momento específico, ou então quando todos os eventos terminam a execução.

As *Scripts* são também executadas sob o controle do *Waf* com a opção *"- -run"* e, a título de exemplo, na Figura 5.2 podemos observar o resultado da execução de uma *script* de teste disponibilizada no simulador.

```
MacBook:ns-3-dev susanasousa$ ./waf --run scratch/myfirst
Waf: Entering directory `/Users/susanasousa/ns-3-allinone/ns-3-dev/build'
Waf: Leaving directory `/Users/susanasousa/ns-3-allinone/ns-3-dev/build'
'build' finished successfully (1.374s)
At time 2s client sent 1024 bytes to 10.1.1.2 port 9
```

Figura 5.2: Execução da *Script* 'MyFirst.cc'

Abstrações usadas pelo NS-3

No simulador *NS-3*, as entidades do mundo real são representadas por abstrações com um significado específico no contexto da simulação. A Figura 5.3 mostra como se relacionam os objetos chave no simulador *NS-3*, sendo de seguida feita uma breve

descrição das abstrações mais relevantes no contexto deste trabalho.

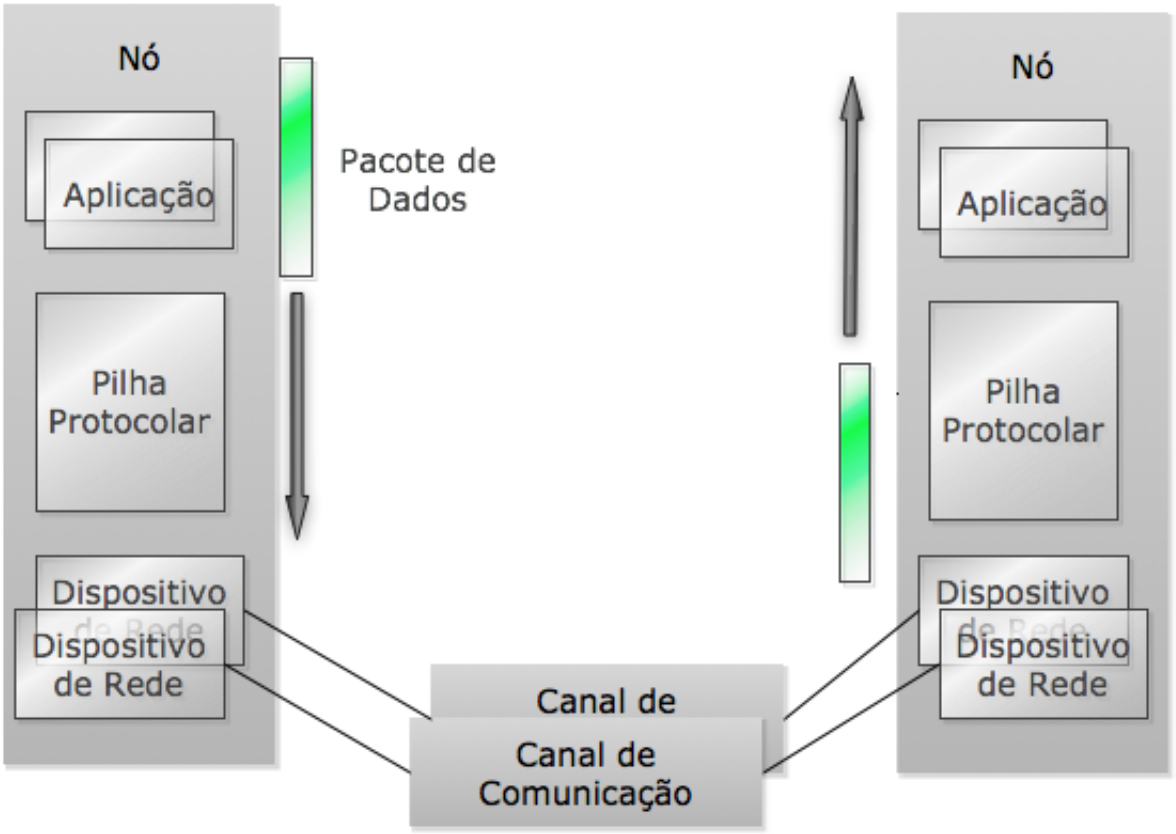


Figura 5.3: Modelo de Objetos NS-3

O *Nó* no *NS-3* é representado pela classe *C++* **Node** que contém todos os métodos que permitem criar e gerir o que habitualmente designamos, nas redes reais de nó, como um encaminhador ou um terminal. A este tipo de objeto, nó, serão adicionadas funcionalidades, como aplicações, protocolos e dispositivos de rede.

Os **NetDevices** do *NS-3* são uma abstração dos dispositivos de rede instalados nos nós. Estes dispositivos vão permitir conetar os diversos nós que compõem uma topologia, através de canais de comunicação. A classe C++ *Netdevice* fornece métodos que fazem a gestão destas mesmas conexões entre objetos *Node* e *Channel*.

As **Queue** do *NS-3* são as representações das filas de espera nos dispositivos de rede. A classe C++ *Queue* do *NS-3* fornece métodos como *DoEnqueue*, *DoDequeue* ou *Drop*, para que seja dado um tratamento adequado aos pacotes de dados que

chegam a um nó. O método *DoEnqueue* é evocado pelo dispositivo de rede para colocar o pacote na fila para ser transmitido, e o método *DoDequeue* é evocado para retirar o pacote da fila e enviá-lo pelo canal de comunicação. A definição do tipo de fila de espera pode variar de acordo com o cenário de simulação.

O canal de comunicação estabelecido entre dois dispositivos de rede é representado no *NS-3* na classe C++ **Channel**. De forma a aproximar o mais possível este simulador da realidade das redes de computadores, o *NS-3* permite a definição de vários tipos de canais de comunicação, desde canais Wi-Fi a canais ponto a ponto, dependendo do tipo de NetDevice. A título de exemplo, entre dispositivos do tipo Wi-Fi, o canal estabelecido será necessariamente um canal Wi-Fi.

A classe C++ **Application** surge no *NS-3* como uma abstração para aplicações dos utilizadores, associadas aos nós, que geram atividades para simulação. Existem vários tipos de aplicações no simulador, desde geradores de tráfego, Clientes e Servidores UDP, Echo, entre outras, que são geridas pelos métodos disponibilizados na classe.

A classe **Packet** do simulador é uma abstração para os pacotes de dados que circulam nas redes reais, e trata-se aqui de uma estrutura de dados com a capacidade de suportar fragmentação e desfragmentação, extensibilidade, suporte para aplicações de dados reais e virtuais, entre outras. No decorrer das simulações os pacotes de dados vão circular pela rede obedecendo a determinados critérios de encaminhamento, entre pares origem/destino.

Dada a complexidade de representar em ambiente de simulação, da forma mais fiel possível, um cenário real, com um determinado número de encaminhadores, dispositivos de rede e conexões, fazendo circular pacotes de dados entre eles, o *NS-3* disponibiliza assistentes de topologia, **Topology Helpers**, que agregam as operações envolvidas no processo de desenho e conceção de uma topologia em modelos mais simples de utilizar.

O *NS-3* disponibiliza um módulo de mensagens, designado de *"Logging"*, para informação de *Debug*, avisos, de erro ou algum tipo de mensagem informativa que se pretenda obter das *Scripts* ou dos modelos. Estas mensagens são apresentadas na linha de comandos no decorrer da simulação e o nível de detalhe é totalmente configurável. Este módulo, considerando a complexidade do simulador, torna-se num excelente aliado no acompanhamento do progresso das simulações.

5.2 Modelo *DiffServ* para o NS-3

O *Network Simulator 3* não inclui nativamente uma implementação para diferenciação de tráfego em redes *IP*, uma alternativa ao modelo atualmente em vigor na *Internet*, o modelo de melhor esforço.

No ano académico de 2010/2011, Sanjay Ramroop desenvolveu, para o *NS-3*, uma arquitetura de diferenciação de tráfego [1] baseada no modelo proposto pelo IETF[14]. A existência deste modelo, disponibilizado publicamente, foi decisiva na escolha do simulador para desenvolver todo o trabalho a nível de diferenciação de tráfego no processo de encaminhamento.

Integração do modelo DiffServ

Na página do projeto desenvolvido por Sanjay Ramroop [1], é disponibilizado o código fonte de todas as classes do modelo, com exceção de duas classes, nomeadamente *"SeqIdTag.cc"* e *"SeqIdQueueTag.cc"*, mas que, tratando-se de adaptações da classe *"FlowIdTag.cc"* do *NS-3*, foram facilmente desenvolvidas.

A integração do modelo passou inicialmente por incluir uma pasta com todas as classes disponibilizadas dentro da zona destinada aos modelos (*ns-3-allinone/ns-3-dev/src/*), e configurar a script de execução do *waf* a *wscript*. Na *wscript* são especificados todos os ficheiros fonte que constituem o modelo, bem como as dependências existentes com outros modelos.

Para testar o funcionamento do modelo, foi usada a script de simulação disponibilizada na página do projeto, *"myDiffServNetwork.cc"*, que foi colocada na pasta destinada aos ficheiros de simulação, *ns-3-allinone/ns-3-dev/scratch/*. Foi neste ponto necessário proceder a algumas alterações no código disponibilizado, pelo facto do modelo ter sido desenvolvido para uma versão antiga do simulador, a versão 3.8, e tratando-se o *NS-3* de um simulador com código fonte aberto, com um elevado número de terceiros a contribuir para o seu desenvolvimento, existirem alterações frequentes na *API* que devem ser consideradas. Depois das devidas alterações e testes efetuados, foi possível simular a cenário desenvolvido por Sanjay Ramroop, com diferenciação de tráfego nos encaminhadores.

O trabalho desenvolvido em [1] assenta, essencialmente, na criação de um novo tipo de filas de espera (*DiffServQueue*), nos dispositivos de rede. A arquitetura introduz, para o *NS-3*, o conceito de domínio *DiffServ*, com distinção entre nós fronteira e nós internos. Os pacotes de dados que circulam na rede recebem um

tratamento diferenciado ao longo do percurso da origem até ao destino. Todo o tráfego gerado pelas aplicações passa a ser mapeado em classes de serviço.

De uma forma genérica, o desenho da arquitetura *DiffServ* divide-se em três áreas fundamentais:

Definição da estrutura dos SLAs O cliente contrata com o provedor de serviços um serviço adequado ao tipo de tráfego que pretende enviar através do domínio de diferenciação, na forma de *SLA*. O *SLA* é usado para determinar o tipo de tratamento que é dado a determinado fluxo no interior do domínio e, para tal, todos os pacotes de dados pertencentes ao fluxo são marcados com o mesmo *DSCP*. O *SLA* especifica o conjunto de aplicações contratadas pelo cliente, a classe de encaminhamento a que os pacotes pertencem, e contém o perfil do tráfego, com valores como taxas de transmissão acordadas ou taxas de pico. Especifica também as ações a serem tomadas no caso dos pacotes não estarem em conformidade com o perfil contratado.

Nós Fronteira Os nós na fronteira do domínio de diferenciação são responsáveis por determinar qual o *SLA* associado ao fluxo a que os pacotes pertencem. Depois de determinarem se existe um *SLA* correspondente na base de dados dos *SLAs*, aplicam um algoritmos de teste para verificar se o pacote se encontra em conformidade com o perfil contratado pelo cliente. Por último procedem à marcação do pacote com o valor determinado nas ações anteriores. A partir daqui assumem as mesmas responsabilidades do que os nós internos.

Nós Internos Os nós no interior do domínio têm como função classificar, em termos de comportamento, os pacotes de tráfego que chegam pelos dispositivos de rede e mediante o resultado desta operação, colocá-los na fila de espera correta. Neste modelo *DiffServ* foram implementados três grupos normalizados de comportamentos, *AF, EF e BE*, cada um deles com mecanismos de escalonamento e gestão próprios, divididos em seis filas de espera.

Filas do modelo *DiffServ*

A fila **EF** (Expedited Forwading), é a fila com maior prioridade. Os pacotes de dados com o valor *DSCP* correspondente à classe *EF*, e que não excedam o perfil de tráfego acordado, são colocados nesta fila. Esta é servida por um algoritmo de gestão de fila *Drop Tail*. De forma a efetivamente atribuir maior prioridade a esta fila é aplicado um algoritmo de escalonamento *Priority Queue* na saída, combinado

com um *Token Bucket* na entrada para limitar as taxas de saída dos pacotes pertencentes à classe.

Nesta implementação estão disponíveis quatro filas **AF**(Assured Forwarding), **AF1**, **AF2**, **AF3** e **AF4**, com diferentes níveis de descarte de pacotes e prioridade no acesso aos recursos da rede. Para atingir diferentes níveis de descarte é aplicado o algoritmo *WRED* (Weighted Random Early Detection), que aplica políticas de descarte de acordo com os requisitos da classe de serviço. Para atribuír prioridades distintas entre as classes *AF* é aplicado um algoritmo de escalonamento *WRR*, com um peso por classe. O maior peso é atribuído à classe *AF1* para que esta tenha a maior prioridade dentro das classes *AF*.Tal como na classe **EF**, os pacotes que entram no nó e vêm marcados com o *DSCP* desta classe de serviço, são colocados na respetiva fila e alvo de tratamento adequado.

A fila **BE** (Best Effort), é a fila por defeito. Para esta fila são escalonados pacotes de dados que não tenham qualquer tipo de marcação no seu cabeçalho ou pacotes que tenham sido alvo de um tipo de ação de conformidade, por não respeitarem o perfil de tráfego acordado. No interior da fila é aplicado o algoritmo de gestão de filas *DropTail*. É combinada com as filas **AF** com algoritmo *WRR*.

Todas as filas são servidas por um algoritmo *FIFO* no seu interior, ou seja, os pacotes são despachados para a rede exatamente pela ordem que são colocados nas filas.

A arquitetura *DiffServ* foi implementada como subclasse da classe *Queue* do *NS-3*, designada *DiffServQueue*. A subclasse disponibiliza, tal como a *Queue*, as funções de *DoEnqueue* e *DoDequeue*, além de todos os mecanismos de diferenciação do modelo.

Na Figura 5.4 retirada de [1], podemos mais facilmente visualizar de que forma os diferentes componentes interagem no processo de diferenciação, e em que ponto são aplicados os diferentes algoritmos implementados.

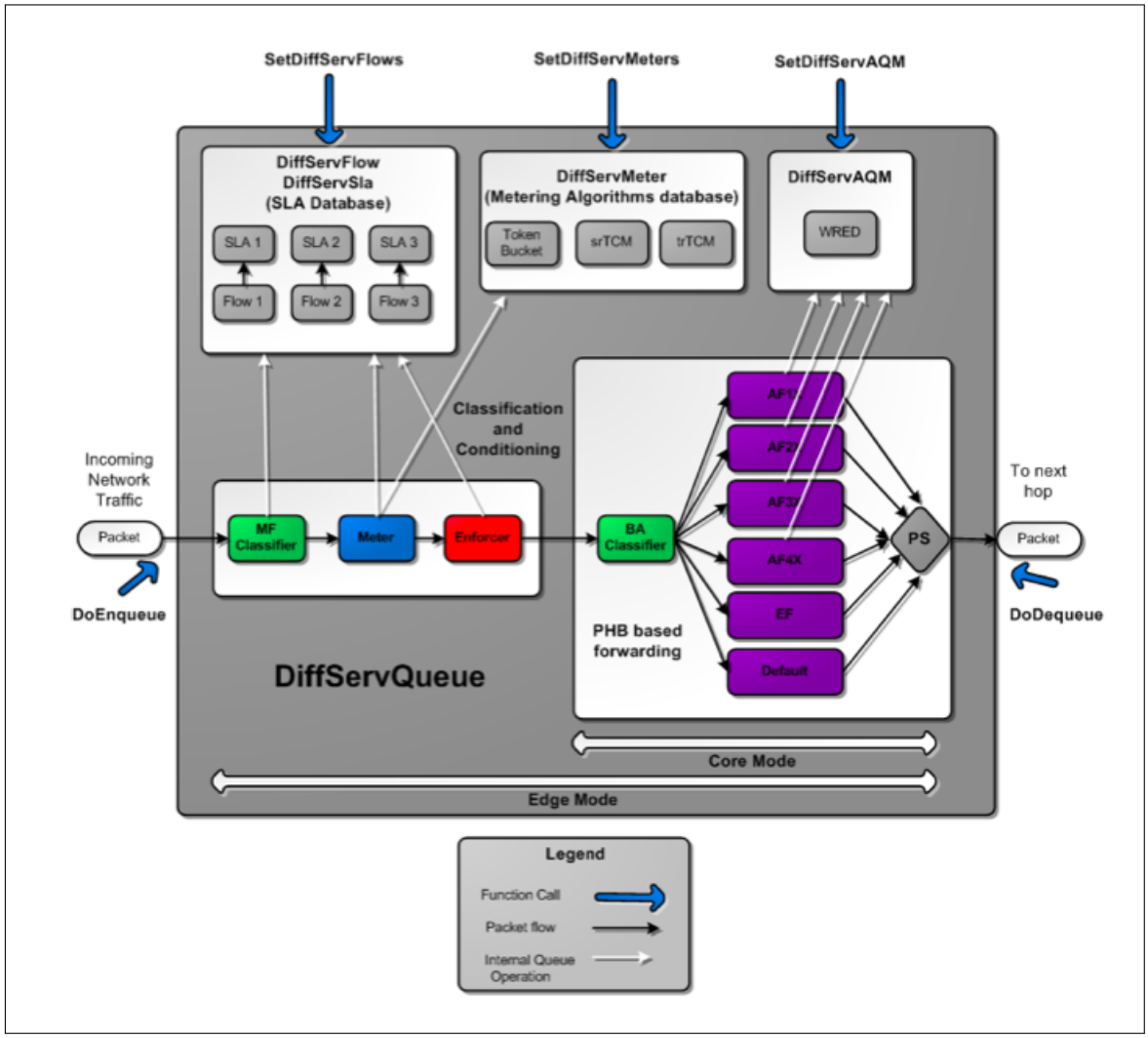


Figura 5.4: Arquitetura DiffServ

Classes do modelo *DiffServ*

O modelo *DiffServ* para *NS-3* é constituído por um conjunto de classes *C++*, cada uma com funções específicas e que permitem que seja possível introduzir diferenciação nas filas dos nós da rede, no contexto de simulação.

A classe principal é a **"DiffServQueue"**, e faz a gestão das filas nos dispositivos de rede de forma diferenciada, dependendo da classe do tráfego. Neste modelo de diferenciação são inicializadas seis filas em cada dispositivo, uma fila *AF1*, uma fila *AF2*, uma fila *AF3*, uma fila *AF4*, uma fila *EF* e uma fila *BE*. A classe *DiffServQueue* pode ser configurada para operar no modo fronteira ou no modo interno do domínio.

A classe **"DiffServFlow"** contém os métodos necessários à configuração e iden-

tificação de um fluxo. A um fluxo de dados é atribuído um identificador único e um apontador para o correspondente *SLA*. Estes valores permitem posteriormente identificar pacotes pertencentes a um fluxo, além de auxiliar na análise do desempenho por fluxo.

Toda a informação dos clientes está contida na classe **"DiffServSla"**, especificamente os perfis de tráfego e algoritmos a serem usados para os diferentes *SLAs*. A classe tem uma interação direta com os nós fronteira, no sentido de fornecer a informação necessária para a classificação e acondicionamento do tráfego que vai entrar no domínio de diferenciação. Nos nós internos vai auxiliar no processo de policiamento e acondicionamento.

A classe abstrata **"DiffServMeter"** auxilia, ao nível dos nós fronteira, na determinação do nível de conformidade dos pacotes de dados com o *SLA* acordado. Divise-se em três algoritmos distintos, representados nas três subclasses *TokenBucket*, *srTCM*, *trTCM*.

Para auxiliar na gestão das filas do tráfego pertencente às classes *AF*, aplicando o algoritmo *WRED*, é usada a classe abstrata **"DiffServAQM"**. Às restantes classes de tráfego é aplicado um algoritmo *DropTail*.

A classe **"StatCollector"** desempenha no modelo *DiffServ* para *NS3* um papel essencial no processo de análise do desempenho tanto das filas de diferenciação como do desempenho de um fluxo fim a fim. No decorrer da simulação é feita a coleção dos dados mais relevantes, como tempos de entrada e saída de pacotes nas filas e número de pacotes descartados. Estes dados permitem determinar valores estatísticos, fundamentais na análise de desempenho do modelo. São gerados dois ficheiros de texto, um com estatísticas por fila, como atraso médio por fila, número total de pacotes colocados em fila e descartados, e um outro ficheiro com estatísticas por fluxo. Nas Figuras 5.5 e 5.6 é apresentado um exemplo do *output* do ficheiro das filas e dos fluxos, respetivamente.

****STATISTICS COLLECTOR****

Total number of DiffServQueues: 54

DiffServQueue number: 0

PHB Queue number: 0--AF1 Queue

PHB Queue Summary:
Total number of packets to be enqueued: 3451
Total number of packets dropped: 191
Percentage of packets dropped: 5.53463%
Queueing Delay 50th percentile: 0.0678368
Average Queue Length : 9.02689

Sequence Numbers	Enqueue Time	Dequeue Time	Queued Time	Packet drop	Current Queue Size	Packet Size
1	0.00952	0.00952	0	0	0	564
2	0.0560533	0.0560533	0	0	0	1054
3	0.0826933	0.112773	0.03008	0	0	548
4	0.0879733	0.136667	0.0486933	0	1	164
5	0.0935733	0.142587	0.0490133	0	2	420
6	0.103973	0.155333	0.05136	0	3	548
7	0.112667	0.164187	0.05152	0	4	420
8	0.144745	0.171333	0.0265883	0	2	116
9	0.154745	0.17288	0.018135	0	3	116
10	0.164745	0.174427	0.00968166	0	2	116
11	0.174745	0.175973	0.00122832	0	0	116
12	0.184745	0.184745	0	0	0	116
13	0.194745	0.194745	0	0	0	116
14	0.204745	0.204745	0	0	0	116
15	0.214745	0.214745	0	0	0	116

Figura 5.5: Filas - StatCollector

****STATISTICS COLLECTOR****

Total number of flows: 12

Flow number: 0
Source IP Address: 10.0.24.1
Destination IP Address: 10.0.23.2
Source Port Number: 49154
Destination Port Number: 5112

Flow Summary:
Number of packets sent: 918
Number of packets dropped: 134
Percentage of packets lost: 14.5969%
Delay 50th percentile: 0.142467
DelayVar 50th percentile: 0.0203333

Sequence Numbers	Tx Time	Rx Time	Delay	Delay Variation	Packet Loss	Packet Size
1	0	0.0476	0.0476	999	0	534
2	0.04	0.120267	0.0802667	0.0326667	0	1024
3	0.04	0.167013	0.127013	0.0467467	0	518
4	0.04	0.1692	0.1292	0.00218666	0	134
5	0.04	0.176347	0.136347	0.00714666	0	390
6	0.24	0.303	0.063	0.0733466	0	765
7	0.24	0.308827	0.0688267	0.00582665	0	407
8	0.24	0.321933	0.0819333	0.0131067	0	504
9	0.36	0.4322	0.0722	0.00973336	0	903

Figura 5.6: Fluxos - StatCollector

5.3 Encaminhamento Global no NS-3

O Simulador *NS-3* tem suporte para alguns protocolos de encaminhamento tradicionais, entre eles o **Encaminhamento Global Centralizado** para *IPv4*, conhecido neste ambiente de simulação por *GOD*. Este protocolo corre um algoritmo de caminho mais curto, construindo desta forma as tabelas de encaminhamento à semelhança do protocolo *OSPF*. O código que implementa o *GOD* é baseado na versão do *software quagga* [37]. Para habilitar o encaminhamento global nos nós de uma topologia, basta incluir no ficheiro de simulação o módulo *Internet*(Listagem 5.1):

Listing 5.1: *include* do módulo *"internet"*

```
(...)
#include "ns3/internet-module.h"
(...)
```

Este módulo garante que é usada por defeito a pilha protocolar da *Internet*, *InternetStackHelper*, e a cada nó é associada uma instância de encaminhamento global (*"GlobalRouter"*). Depois de atribuídos os endereços *IPv4* e configuradas as ligações entre os nós, para que sejam efetivamente construídas as tabelas de encaminhamento evocamos a seguinte função:

Listing 5.2: Função para calcular tabelas de encaminhamento

```
(...)
Ipv4GlobalRoutingHelper::PopulateRoutingTables();
(...)
```

Apesar de apenas ser necessário incluir no ficheiro de simulação estas duas linhas de código para habilitar o encaminhamento global e construir as tabelas de encaminhamento, a sua implementação envolve bastantes componentes e não só do modelo *"internet"*. De seguida será explicado, passo a passo, o funcionamento do protocolo:

1. Numa primeira fase são construídas as tabelas de encaminhamento vazias. Um objeto *GlobalRouteManager*, utilizando a *API* (Application Programming Interface) *IPv4*, percorre a lista de nós da topologia, para encontrar os interfaces com o objeto *GlobalRouter* associado. Aos nós que participam é associado o objeto *Ipv4GlobalRouting*.

2. O mesmo objeto *GlobalRouteManager*, num ciclo que percorre todos os nós, faz o pedido dos *LSAs*(Link State Advertise) ao objeto *GlobalRouter* que, por

sua vez, inicia a descoberta dos nós adjacentes e custos de ligação associados: Numa primeira fase cada nó origina um *LSA*, de seguida são determinados o número de interfaces que o nó contém e o número de interfaces que estão ligados ao nó, que não são necessariamente ligações a outros nós (por exemplo ethernets). São efetuados alguns testes, nomeadamente para determinar o tipo de canal que liga dois interfaces, e qual o tipo de interface adjacente ao nó. E se se tratar de um interface que cumpra os requisitos e limitações do protocolo, através do objeto *Ipv4* associado ao nó, será obtida toda a informação necessária para construir o *LSA*, que será depois inserido na base de dados topológica.

3. Neste ponto terá início a segunda fase do processo, a inicialização das rotas. Para cada um dos nós que participam no encaminhamento global, é executado o algoritmo de caminho mais curto *SPF* usando a base de dados dos *LSAs*. Com os caminhos obtidos, são preenchidas as tabelas de encaminhamento.

Com as tabelas construídas, cada nó será agora responsável por selecionar a melhor rota para os pacotes de dados que vão circular na rede durante a simulação. Para esse efeito, a classe **Ipv4RoutingProtocol** contém dois métodos para determinar as melhores rotas, que são:

- **RouteOutPut** Para pacotes de dados que viajam para o exterior a partir de um terminal. O pedido de rota é feito diretamente pela camada de transporte, e como resposta é-lhe retornado um apontador para o objeto *Ipv4Route*, que é também passado para a camada de rede (para evitar nova pesquisa). Na Figura 5.7 podemos visualizar os objetos envolvidos no processo de pedido de uma rota do tipo *RouteOutPut*.

- **RouteInPut** Para os pacotes de dados que chegam pelos interfaces dos encaminhadores. O pedido de rota é feito pela camada de rede, especificamente pelo objeto *Ipv4L3Protocol*, e neste caso a resposta vai depender do destinatário final do pacote. Desta forma, é associado ao pedido de rota um *Callback*, que se trata de um mecanismo intrínseco ao *NS-3* e, neste caso específico, auxilia a determinar se o pacote se destina a um grupo multicast, a um nó específico ou mesmo ao nó que está a requisitar a rota. Na Figura 5.7 está também representado o processo de pedido de uma *RouteInPut*.

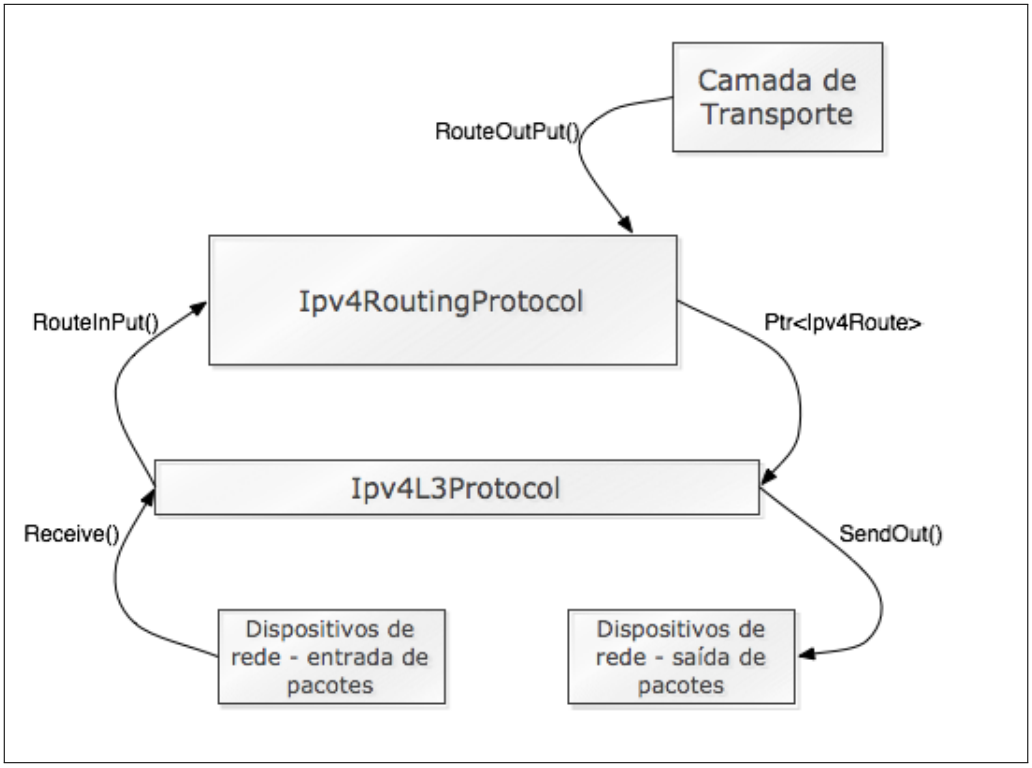


Figura 5.7: Pedido de Rota

5.4 Adaptação do protocolo de Encaminhamento GOD

Depois de integrado o modelo de diferenciação, garantindo que passávamos a poder instalar nos dispositivos de rede filas do tipo *"DiffServQueue"*, foi necessário alterar o protocolo de encaminhamento para que passasse a ter em consideração as classes de serviço dos fluxos de tráfego gerado pelas aplicações. As tabelas de encaminhamento dos encaminhadores que participam ativamente no encaminhamento global passam a ter o caminho mais curto para cada destino, por cada uma das classe de serviço.

Alteração dos *LSAs*

O funcionamento do protocolo, como foi descrito atrás, pressupõe que numa primeira fase seja efetuado o pedido a cada um dos encaminhadores dos *LSAs* respetivos, que contêm a informação do estado das suas ligações. A primeira alteração foi então realizada ao nível da construção dos *LSAs*, que para as ligações do tipo

ponto a ponto passavam a ter informação do custo de ligação de cada classe de serviço.

Na classe *"global-router-interface.cc"* do módulo *"internet"* do *NS-3* é feita a descoberta dos *SLAs* representados na classe C++ por objetos *"GlobalRoutingLSA"*, que contêm a informação do custo das ligações, objetos *"GlobalRoutingLinkRecord"*.

Na Listagem 5.3 podemos visualizar um excerto relativo ao processamento de uma ligação do tipo *ponto a ponto* onde é feito o pedido do custo da ligação percecionado por cada uma das classes de serviço.

Listing 5.3: Método *"ProcessPointToPointLink"*

```
void
GlobalRouter::ProcessPointToPointLink (Ptr<NetDevice> ndLocal,
   GlobalRoutingLSA *pLSA)
{
(...)
  //Passo a invocar o método GetMetric do objeto ipv4Local com + 1
      parâmetro, o ndLocal(NetDevice)
  uint16_t metricLocal = ipv4Local->GetMetric (interfaceLocal,
     ndLocal, 0); //AF1
  uint16_t metricLocal1 = ipv4Local->GetMetric (interfaceLocal,
     ndLocal, 1);//AF2
  uint16_t metricLocal2 = ipv4Local->GetMetric (interfaceLocal,
     ndLocal, 2);//AF3
  uint16_t metricLocal3 = ipv4Local->GetMetric (interfaceLocal,
     ndLocal, 3);//AF4
  uint16_t metricLocal4 = ipv4Local->GetMetric (interfaceLocal,
     ndLocal, 4);//EF
  uint16_t metricLocal5 = ipv4Local->GetMetric (interfaceLocal,
     ndLocal, 5);//BE
(...)
}
```

O conteúdo do novo objeto *"GlobalRoutingLSA"* contruído, baseado na estratégia proposta, pode ser observado quando habilitamos as mensagens de *Logging* no decorrer da simulação, especificamente na classe *"global-router-interface.cc"*.

Na Figura 5.8 é apresentada uma parte do *SLA* de um dos encaminhadores, onde podemos identificar a informação de uma ligação do tipo *ponto a ponto*, com os diferentes custos percecionados por cada classe de serviço, e uma outra ligação do tipo *stub* onde apenas é referenciado um custo único.

```
---------- RouterLSA Link Record ----------
m_linkType = 1 (GlobalRoutingLinkRecord::PointToPoint)
m_linkId = 0.0.0.3
m_linkData = 10.0.6.1
m_metric = 1
m_metric1 = 1
m_metric2 = 1
m_metric3 = 10
m_metric4 = 1
m_metric5 = 1
---------- End RouterLSA Link Record ----------
---------- RouterLSA Link Record ----------
m_linkType = 3 (GlobalRoutingLinkRecord::StubNetwork)
m_linkId = 10.0.6.2 (Network number of attached network)
m_linkData = 255.255.255.0 (Network mask of attached network)
m_metric = 1
---------- End RouterLSA Link Record ----------
```

Figura 5.8: *SLA* do encaminhador

Custos das ligações por classe de serviço

Se considerarmos o funcionamento nativo do protocolo de encaminhamento global do *NS-3*, o pedido do custo de uma ligação é feito ao objeto *C++ "ipv4Local"* que representa um apontador para a interface do encaminhador, e retorna um custo fixo que não se altera durante a simulação, a não ser que a ligação falhe por algum motivo.

Na estratégia adotada, como podemos observar na Listagem 5.3, é enviado como parâmetro ao objeto *"ipv4Local"*, além do identificador do interface, o identificador do dispositivo de rede *ndLocal* e a classe de serviço associada ao custo que pretendemos obter. Com o identificador do dispositivo de rede conseguimos determinar a fila para onde são direcionados os pacotes, que no caso de se tratar de uma fila do tipo *DiffServ* deve ser analisada para se obterem os custos da ligação associados a cada classe.

Na Listagem 5.4 podemos observar o método da classe *"ipv4-interface.cc"* do módulo *"internet"*, onde é feito o pedido de custo de ligação à classe *"StatCollector.cc"* do modelo de diferenciação usado.

Listing 5.4: Método *"GetMetric"*

```
uint16_t Ipv4Interface::GetMetric (Ptr< NetDevice > ndLocal, int
    perhop)
{
  uint_16 id, metr;
```

```
    uint_16 phb = perhop;
    Ptr<Queue> queue;
    Ptr<NetDevice> dev = ndLocal;

    //DimanicCast de apontador NetDevice para PointToPointNetDevice
    Ptr<PointToPointNetDevice> p2pdev = DynamicCast<
        PointToPointNetDevice> (dev);

    if(p2pdev==0){ //Não p2p
      metr = 1;
    }
    else{ // p2p
      queue = p2pdev->GetQueue();

      //Cast para DiffServQueue
      Ptr<DiffServQueue> dq = DynamicCast<DiffServQueue> (queue);
      if(dq==0){ // não é queue DiffServ
        metr = 1;
      }
      else{ //Se a fila for DiffServ
      //Determina id da Fila,
        id = dq->GetQueueId();
        if(phb == 5){ //BE
          metr = 1;
        }
        else{
          Ptr<StatCollector> statCollector;
          metr = statCollector->Metric(id, phb);
        }
      }
    }
    return metr;
  }
```

Aqui, como podemos observar, o método da *"StatCollector.cc"* é evocado no caso de se tratar de uma ligação ponto a ponto (limitação do modelo *DiffServ* utilizado), com fila do tipo *"DiffServQueue"*(tem diferenciação nas flias), e para todas as classes excepto a classe *BE* que tem um custo de ligação associado de valor fixo, baseado no número de saltos.

Atualização e Cálculo dos custos das ligações

A classe *"StatCollector.cc"* do modelo de diferenciação [1] desempenha na estratégia implementada duas funções essenciais. Por um lado faz a monitorização das

filas *"DiffServQueue"* nos encaminhadores para determinar o estado de congestionamento da rede e, por outro lado, a partir da informação da monitorização efetua o cálculo dos novos custos das ligações, associados às classes de serviço.

Tal como referido no capítulo do desenho da implementação, a classe *"StatCollector.cc"* é responsável por fazer a recolha dos tempos de entrada e saída dos pacotes de dados nas filas dos dispositivos de rede. No momento em que uma fila do tipo *"DiffServQueue"* é criada, a classe *"StatCollector.cc"* atribui-lhe um identificador único e cria um conjunto de vetores onde, no decorrer da simulação, armazena toda a informação dos pacotes que passam na fila de forma sequencial. Sempre que um pacote entra ou sai de uma das filas *"DiffServQueue"*, é evocado um método da *"StatCollector.cc'*, *"StatCollector::Enqueue"* ou *"StatCollector::Dequeue"*, para que sejam calculados e armazenados os parâmetros alvo de monitorização.

No modelo de diferenciação original, o resultado da monitorização é apresentado num ficheiro de texto apenas no final da simulação. No contexto da estratégia desenhada, as filas são alvo de monitorização periódica, num intervalo definido por configuração, no decorrer da simulação. Os resultados obtidos em cada monitorização são comparados com os resultados obtidos na monitorização anterior, que são guardados em vetores criados para o efeito. Desta forma, sempre que é evocado o método *"StatCollector::OutputQueueStats"*, é analisado o desempenho de cada uma das filas dos dispositivos de rede de acordo com os parâmetros referidos no desenho da estratégia, percentagem de pacotes perdidos para as filas de tráfego *AF* e atraso médio para as filas de tráfego *EF*, e comparados esses valores com os valores armazenados da monitorização anterior. No caso de em alguma das comparações efetuadas nas filas do domínio de diferenciação, ser determinada a necessidade de proceder a uma atualização dos custos, então é evocado o método do protocolo de encaminhamento global para proceder à reconstrução das tabelas de encaminhamento.

Na Listagem 5.5 podemos observar a operação efetuada na classe *"StatCollector.cc"*, especificamente na análise do desempenho da fila de tráfego *EF*.

Listing 5.5: Monitorização da fila de tráfego *EF*

```
(...)
  if((j == 4)){

    //verifica a diferençao entre valores anterirores e atuais
    valD = fabs((holdDelay.at(i).at(j)*1000) - ((averageQueuedTime
       )*1000));
```

```
      //se for superior ao termo de comparação
      if(valD >= 1.0){

        //atualiza as tabelas de encaminhamento
        Ipv4GlobalRoutingHelper::RecomputeRoutingTables();
      }
    }
  (...)
```

É também na classe *"StatCollector.cc"* que são calculados os custos das ligações percecionados por cada uma das classes de serviço. Sendo estes custos baseados nos requisitos de *QoS* das aplicações, são usados os valores obtidos na monitorização, a percentagem de pacotes perdidos para as classes *AF* e o atraso médio para a classe *EF*. Para este efeito foi criado um novo método *"StatCollector::Metric"* na classe, que é evocado pelo protocolo de encaminhamento global do simulador, como foi apresentado na Figura 5.10.

O novo método, tal como o método de monitorização, percorre todos os vetores onde são armazenados os tempos de entrada e saída de pacotes nas filas dos encaminhadores, para calcular os parâmetros pretendidos desde a última atualização dos custos.

Na Listagem 5.6 podemos, a título de exemplo, observar o estabelecimento do custo de uma ligação percepcionado pela classe de serviço *EF*, em que o custo é definido no parâmetro *"metr"*.

Listing 5.6: Cálculo do custo da ligação para a classe de serviço *EF*

```
(...)
  //classe de serviço EF
  else if(phb == 4){
    //atraso médio desde a última atualização
      averageQueuedTime = totalQueueTime / queueCount;

    //no caso de não existirem valores de delay
    if(isnan(averageQueuedTime == 0 ){
      //igual ao custo mínimo
      metr = 1;
    }
    else{
      //igual a função do atraso médio na fila
      metr = ((int)(averageQueuedTime*1000);
    }
```

```
  }
(...)
```

O custo é igual a 1, para o caso de não existirem valores de atraso, ou igual a uma função do atraso médio dos valores registados naquela fila. É utilizada uma função do atraso médio, para que os custos de ligação das diferentes classes estejam todos aproximados da unidade, custo de um salto, compatível com o funcionamento do algoritmo de caminho mais curto *SPF*.

Monitorização por Fluxo Da mesma forma que procede para armazenar as informações das filas *"DiffServQueue"*, a classe *"StatCollector.cc"* armazena a informação fim a fim dos fluxos de dados. Quando um fluxo de tráfego é gerado na aplicação do emissor, o protocolo da camada de transporte evoca um método da classe *"StatCollector.cc"* com alguns parâmetros relevantes no processo de monitorização fim a fim.

Na Listagem 5.7 podemos observar a alteração efetuada nos métodos *"Send"* e *"Receive"* do protocolo de transporte *UDP*. O método *"PrepareTx"* da classe *StatCollector* é evocado quando um pacote de dados sai do emissor e o método *"PrepareTx"* é evocado quando um pacote de dados chega ao receptor. O mesmo tipo de alteração deve ser efetuada em todos os protocolos de transporte usados pelas aplicações geradoras de tráfego no âmbito da simulação.

Listing 5.7: Método *"UdpL4Protocol::Send"*

```
(...)
void
UdpL4Protocol::Send (Ptr<Packet> packet,
                     Ipv4Address saddr, Ipv4Address daddr,
                     uint16_t sport, uint16_t dport)
{
  // transmissão de pacote
  StatCollector::PrepareTx(packet, saddr, daddr, sport, dport);
}
(...)
UdpL4Protocol::Receive (Ptr<Packet> packet,
                        Ipv4Header const &header,
                        Ptr<Ipv4Interface> interface)
{
    //recepcao de pacote
    StatCollector::RecordRx(packet);
}
(...)
```

Esta modificação no método do protocolo de transporte não altera o funcionamento do protocolo de encaminhamento, apenas é usada para estudar posteriormente o desempenho fim a fim de cada um dos fluxos que vai ser gerado no contexto da simulação. Com os tempos de entrada e saída dos pacotes de dados na rede de diferenciação, é possível calcular o atraso fim a fim com percentil 50, percentagem total de pacotes perdidos ou variação do atraso. Estes são parâmetros chave na análise ao desempenho da estratégia proposta, especialmente a **Percentagem de perda de pacotes média** e o **Atraso com percentil de 50** que representa a mediana do atraso dos fluxos, ou seja 50% dos valores de atraso fim a fim dos fluxos são iguais ou inferiores à mediana.

Cálculo e inicialização das rotas

Depois de definida a estrutura dos novos *LSAs* e alterado o processo de pedido de custo de ligação, foi necessário alterar a classe *"global-route-manager-impl.cc"* do *NS-3*, onde são calculadas e inicializadas as rotas dos encaminhadores para todos os possíveis destinos.

Dentro do ciclo que percorre todos os encaminhadores da topologia, foi criado um novo ciclo que percorre, para cada encaminhador, todas as classes de serviço, sendo o algoritmo de *Dijkstra* aplicado à classe, como podemos observar no excerto de código apresentado na Listagem 5.8.

Listing 5.8: Caminho mais curto por Classe de Serviço

```
(...)
      if (rtr && rtr->GetNumLSAs () )
        {
    for(int classe = 0; classe < numClasses; classe++){
              SPFCalculate (rtr->GetRouterId (), classe);
    }
        }
(...)
```

Nesta estratégia foi definido que seria possível diferenciar entre seis classes de tráfego, assim cada tabela de encaminhamento passa a ter seis entradas por cada destino, uma por classe.

Para efetivar este comportamento por parte do protocolo, foram feitas algumas alterações na classe *"ipv4-global-routing.cc"* do módulo *"internet"*. Na Listagem 5.9 é apresentado um excerto de código de um dos métodos responsáveis por inserir as

rotas nas tabelas de encaminhamento, o *"Ipv4GlobalRouting::AddHostRouteTo"*.

Listing 5.9: Inserção de rota na tabela de encaminhamento

```
(...)
void Ipv4GlobalRouting::AddHostRouteTo (Ipv4Address dest,
    Ipv4Address nextHop,
                                                uint32_t interface,
                                                    int classe,
                                                    uint32_t distance
                                                    )
{
    Ipv4RoutingTableEntry *route = new Ipv4RoutingTableEntry ();
     *route = Ipv4RoutingTableEntry::CreateHostRouteTo (dest,
         nextHop, interface, classe,
             distance);
     m_hostRoutes.push_back (route);
}
(...)
```

Como podemos verificar, além dos endereços do destinatário e do próximo salto, e do identificados do interface de saída, são acrescentados à rota a classe de serviço e custo total do caminho. Este último parâmetro foi acrescentado à tabela de encaminhamento para que nela conste o custo fim a fim das rotas.

Encaminhamento de pacotes

A inclusão deste novo parâmetro *classe*, no processo de cálculo dos caminhos mais curtos, alterou significativamente o funcionamento do protocolo, especialmente ao nível do encaminhamento de pacotes. Quando um pacote chega ao encaminhador por um dos seus dispositivos de entrada, é analisado o seu cabeçalho *IP* para determinar o destinatário, e a classe a que o pacote pertence, no sentido de selecionar na tabela de encaminhamento o melhor interface de saída para encaminhar o pacote de acordo com a classe de serviço.

Para cumprir o objetivo proposto, foi alterado o método de análise dos pacotes que entram, o *"Ipv4GlobalRouting::RouteInPut"*, e o método que procura o melhor caminho para um destino na tabela de encaminhamento, o *"Ipv4GlobalRouting::LookupGlobal"*. Nas Listagens 5.10 e 5.11 podemos observar excertos de código dos métodos citados, respetivamente.

Listing 5.10: Análise dos pacotes que chegam ao encaminhador

```
(...)
bool Ipv4GlobalRouting::RouteInput  (Ptr<const Packet> p, const
    Ipv4Header &header,
                 Ptr<const NetDevice> idev, UnicastForwardCallback
                    ucb,
                 MulticastForwardCallback mcb,LocalDeliverCallback
                    lcb,
                      ErrorCallback ecb)
{
  //campo tos do cabeçalho
  int tos = header.GetTos();
(...)
```

Listing 5.11: Teste para verificar a classe

```
(...)
      // teste para verificar a classe
      if ((*i)->GetDest ().IsEqual (dest) && clasH == classe)
        {
(...)
```

A principal alteração no método responsável por analisar o cabeçalho *IP* dos pacotes que entram, foi a inclusão de uma função para determinar a classe marcada no campo *"Tos´´* do pacote de dados, como podemos ver na Listagem 5.9. No método de escolha do caminho mais curto, Listagem 5.10, além do destinatário é utilizada a classe de serviço para determinar o melhor caminho.

Para testar o funcionamento dos métodos apresentados foram ativadas as mensagens de *Logging* da classe *"ipv4-global-routing.cc"*. Na Figura 5.9 podemos observar as mensagens que aparecem na linha de comandos quando é adicionada uma nova rota a uma das tabelas de encaminhamento. Podemos também identificar a associação feita a uma das classes de serviço a partir do campo *Tos* do cabeçalho *IP* do pacote, e o processo de procura do melhor caminho por classe de serviço.

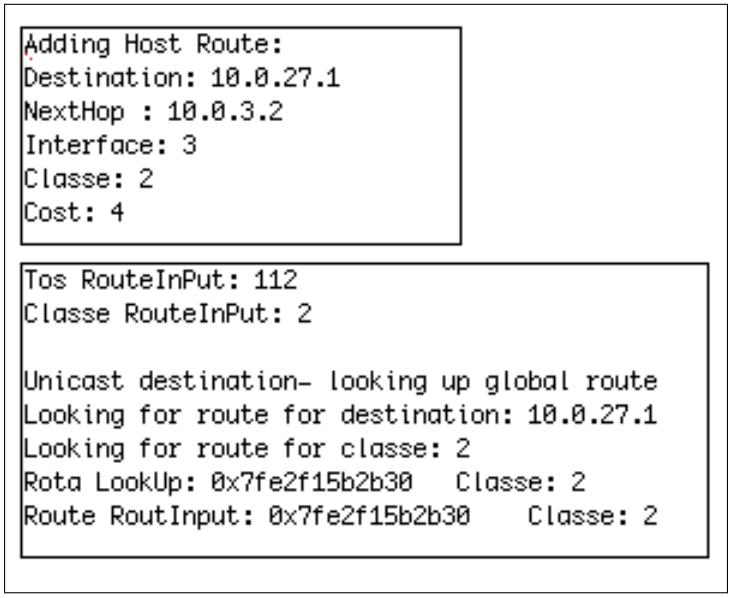


Figura 5.9: Mensagens Logging adição e escolha de rota

Capítulo 6

Testes e Resultados

Neste capítulo serão apresentados os resultados do processo de validação do modelo de encaminhamento de tráfego por classes de serviço, desenvolvido na presente dissertação.

No decorrer dos testes realizados, foram avaliados alguns parâmetros de desempenho, nomeadamente percentagem de perda de pacotes e atraso fim a fim, e efetuada a comparação entre o modelo proposto e outros modelos de encaminhamento sem diferenciação. Os testes foram realizados no simulador de redes *NS-3*, utilizando uma topologia desenvolvida para o efeito pretendido.

6.1 Topologia de Rede

Para testar a implementação do modelo de encaminhamento com diferenciação, foi desenvolvida uma *script C++*, onde é simulada a seguinte topologia de testes, Figura 6.1:

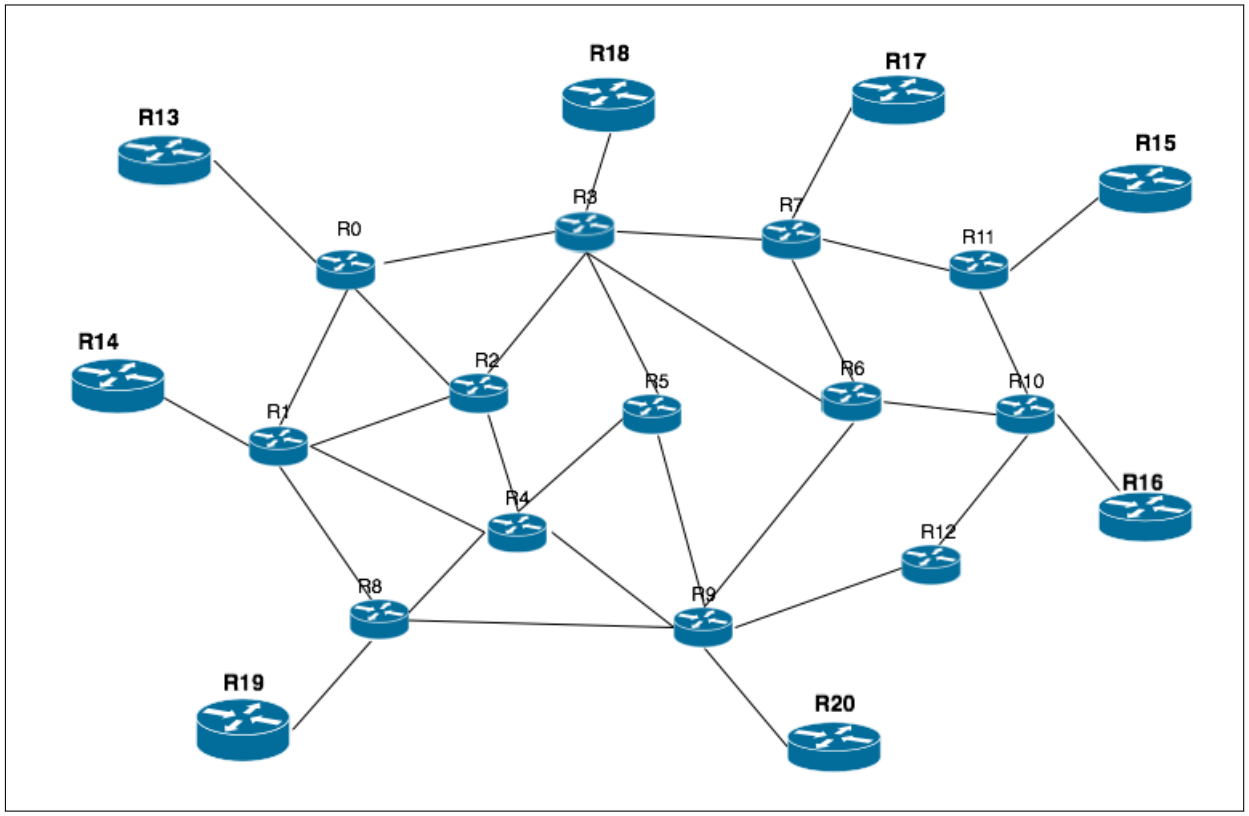


Figura 6.1: Topologia de Rede

Trata-se de uma rede típica *ISP* constituída por 21 encaminhadores, com múltiplos canais de comunicação ponto a ponto entre si, o que se traduz num número de caminhos alternativos fim a fim bastante significativo. Os 8 encaminhadores em destaque na figura, R13, R14, R15, R16, R17, R18, R19 e R20 são os encaminhadores fronteira do domínio *DiffServ*, onde é feita a marcação dos pacotes de dados e neles, de forma aleatória, foram instaladas as aplicações geradoras de tráfego.

Os dispositivos de rede dos encaminhadores são do tipo ponto a ponto, designados no contexto de simulação por *Point-To-Point-NetDevice*, bem como os canais de comunicação estabelecidos entre eles, os *Point-To-Point-Channel*. Todos os canais de comunicação foram configurados com um atraso de propagação constante de *2ms* e uma taxa de transmissão na ordem das centenas de milhares de *bps*. Para os cenários de simulação apresentados neste capítulo foram usados, especificamente, *600000bps* em todos os canais, com exceção dos canais que ligam aos encaminhadores de fronteira, que têm um taxa de transmissão de *10000000bps*. Os canais que ligam aos encaminhadores fronteira têm um limite de taxa de transmissão superior pelo facto de não possuírem interfaces de saída alternativos (Figura 6.1), para garantir que não são criados pontos de congestão que possam afetar o desempenho de

toda a rede, fenómeno designado em inglês *"bottleneck"*.

Para simular as filas nos dispositivos de rede dos encaminhadores, foram configurados dois tipos de filas, as *Red Queue* nativas do simulador [36] e as *DiffServ Queue* de [1]. As filas *RED* são utilizadas num contexto de simulação onde não existe qualquer diferenciação do tráfego que circula na rede, as segundas filas são implementadas nos contextos de simulação onde existe diferenciação por classes de serviço nos encaminhadores. Todas as filas foram configuradas com um limite de 100 pacotes por fila.

6.2 Aplicações geradoras de tráfego

Em todas as simulações de teste realizadas, foram utilizadas aplicações geradoras de tráfego *CBR*, e o simulador NS-3 inclui um módulo *"applications"* onde está disponível um gerador de tráfego deste tipo, designado *OnOff-Application*.

Neste modelo, o tráfego é gerado para um único destino, de acordo com um padrão *OnOff*, ou seja, o gerador de tráfego vai alternando entre os estados 'ligado' e 'desligado', parâmetros configuráveis na script de simulação. Nos testes realizados, apresentados ao longo deste capítulo, as aplicações geram permanentemente tráfego *CBR*, pelo que o estado "desligado" vai ser nulo, como podemos observar no excerto de código apresentado na Listagem 6.1.

Listing 6.1: Aplicação geradora de Tráfego *CBR*

```
(...)
Address sinkAddress15 = InetSocketAddress(addrD15,5881);
PacketSinkHelper sinkHelper15 ("ns3::UdpSocketFactory",
   sinkAddress15);

ApplicationContainer serverApps15 = sinkHelper15.Install(nodes.Get
   (dest15));
serverApps15.Start (Seconds (0));
serverApps15.Stop (Seconds (30.0));


OnOffHelper onoffhelper15("ns3::UdpSocketFactory",sinkAddress15);
onoffhelper15.SetAttribute("MaxBytes", UintegerValue (774172));
```

```
onoffhelper15.SetAttribute("PacketSize", UintegerValue (172));
   //(172)
onoffhelper15.SetAttribute("DataRate", DataRateValue (275200));
   //(68800)
onoffhelper15.SetAttribute("OnTime",StringValue("ns3::
   ConstantRandomVariable[Constant=1.0]")); //(Mean=0.352)
onoffhelper15.SetAttribute("OffTime",StringValue("ns3::
   ConstantRandomVariable[Constant=0.0]")); //(Mean=0.65)

ApplicationContainer clientApps15 = onoffhelper15.Install(nodes.
   Get(src15));
clientApps15.Start (Seconds (0));
clientApps15.Stop (Seconds (30.0));
(...)
```

A aplicação cliente geradora de tráfego é instalada no encaminhador de origem recorrendo à classe *OnOffHelper*. Aqui são configurados atributos como: tamanho dos pacotes de dados, taxa de transmissão em *bps*, número máximo de bytes a transmitir, os tempos a *On* e *Off*, e o protocolo da camada de transporte combinado com o endereço IPv4 e porta do destinatário. No encaminhador destino é instalada a aplicação receptora pelo *PacketSinkHelper*, onde é configurado o mesmo protocolo da camada de transporte especificado na aplicação cliente. Por último, basta definir os instantes em que as aplicações iniciam e terminam. Na Listagem 6.1 podemos observar que ambas as aplicações iniciam no instante 0 e terminam 30 segundos depois.

6.3 Cenários de simulação

Para testar o desempenho da estratégia de encaminhamento por classes de serviço proposta no âmbito desta dissertação, foram alvo de testes 4 cenários distintos: um cenário de encaminhamento sem diferenciação, um cenário de encaminhamento com diferenciação de tráfego nas filas dos encaminhadores da rede, um cenário com diferenciação apenas ao nível do encaminhamento, e um cenário com diferenciação nas filas dos encaminhadores e ao nível do encaminhamento, este último cenário relativo à estratégia proposta.

Para cada cenário foi executado um conjunto de 30 simulações independentes, cada uma com 15 fluxos de tráfego *CBR* entre pares origem/destino escolhidos aleatoriamente entre os encaminhadores de fronteira.

As 30 simulações foram repetidas, em cada cenário, para taxas de transmissão

de *68800bps*, de *137600bps* e de *275200bps*. O objetivo passava por verificar o comportamento da rede, em cada um dos cenários, perante diferentes cargas de tráfego. Considerando o limite das ligações entre os nós internos, *600000bps*, nas situações de baixa carga da rede são necessários 9 fluxos de dados para exceder o limite da ligação, para carga média bastam 5 fluxos e com uma carga alta, representada aqui com taxas de transmissão dos fluxos de *275200bps*, bastam 3 fluxos para que seja excedido o limite do canal de comunicação entre os encaminhadores.

Nos cenários com diferenciação os acordos entre *ISP* e clientes são explicitados na *Script* sob a forma de *SLAs*, e cada fluxo de dados é associado de forma aleatória a uma das 6 classes de serviço. Isto significa que, em média, nas 30 simulações, a quantidade total de tráfego gerada é distribuída de forma uniforme pelas classes de tráfego. Foi tomada a opção de fazer uma distribuição deste tipo, ao contrário do que acontece tipicamente num contexto real, em que a quantidade de tráfego *BE* é superior ao tráfego gerado pelas restantes classes, com o objetivo de criar cenários extremos, onde existe uma quantidade muito elevada de tráfego *QoS* a circular na rede.

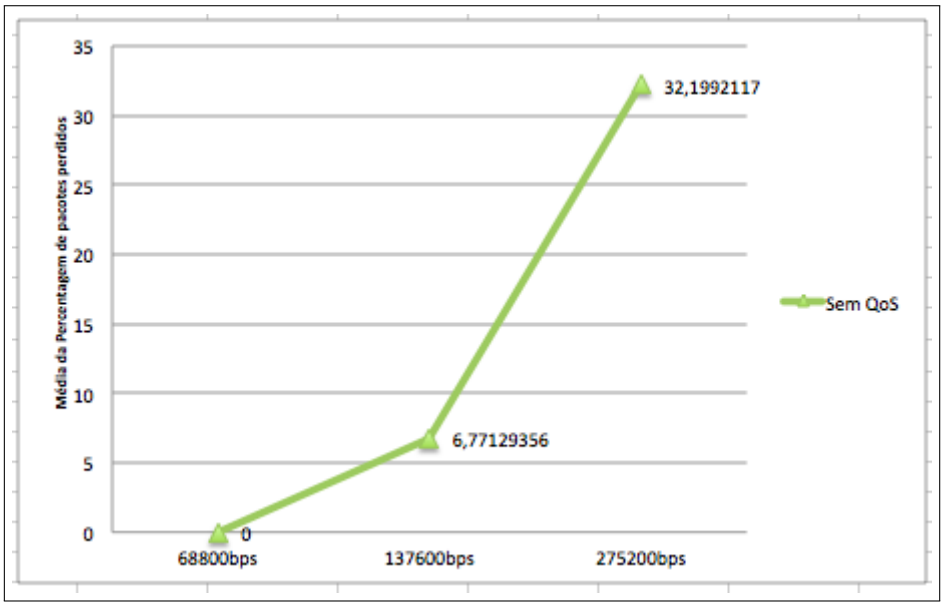
Em todos os cenários foi determinado o nível de desempenho da rede, dadas as condições pré-definidas, em termos de percentagem de pacotes perdidos e atraso fim a fim. A classe *StatCollector* do modelo de diferenciação de [1] é responsável por fazer a coleção dos tempos de entrada e saída dos pacotes e calcular os parâmetros de desempenho, baseados nos valores registados. Os resultados finais apresentam-se graficamente com os valores médios das 30 simulações, para cada uma das classes de serviço.

6.3.1 Cenário 1: Encaminhamento sem diferenciação

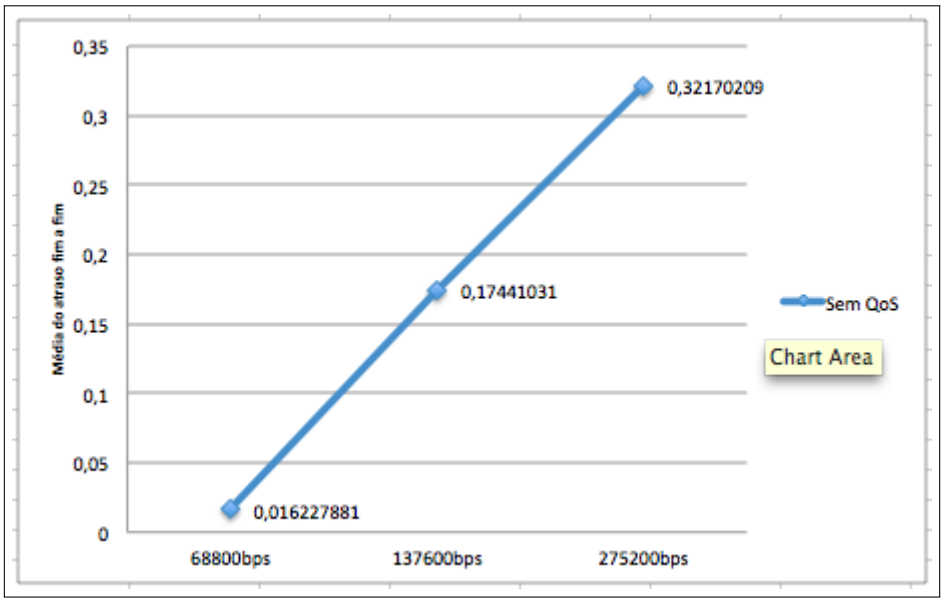
De forma a simular o desempenho da rede quando é aplicado um cenário onde não existe qualquer tipo de diferenciação do tráfego gerado pelas aplicações, foi utilizado o modelo de encaminhamento baseado no serviço de melhor esforço da *Internet*, habilitando o protocolo de encaminhamento global centralizado nativo do *NS-3*.

Depois de configura a *script* de acordo com os pressupostos supracitados, foram realizados os 3 conjuntos de 30 simulações, com taxa de transmissão distinta por conjunto, respetivamente *68800bps*, *137600bps* e *275200bps*. Considerando que neste cenário não existe diferenciação de tráfego por classe de serviço, os resultados apresentados na Figura 6.2 correspondem aos valores médios de percentagem de pa-

cotes perdidos e ao atraso fim a fim com percentil de 50, de todos os fluxos para cada conjunto de simulações.



(a) Média da percentagem de pacotes perdidos



(b) Média do atraso fim a fim

Figura 6.2: Resultados obtidos no cenário sem diferenciação

Os resultados observados nos gráficos estão em concordância com as expectativas. Trata-se de uma estratégia de encaminhamento baseada no caminho mais curto, onde o tráfego da mesma origem para o mesmo destino percorre um caminho idêntico. Considerando o limite dos canais previa-se uma percentagem de perda de

pacotes média elevada, em proporcionalidade com a quantidade de tráfego gerada, Figura 6.2(a). Relativamente à média do atraso, Figura 6.2(b), verificamos que a mesma aumenta em proporcionalidade com a carga de tráfego que percorre a rede.

Este cenário será usado como termo de comparação com os cenários abordados nas secções seguintes, no sentido de verificar em que medida a introdução de mecanismos de Qualidade de Serviço pode otimizar o desempenho da rede, quer do ponto de vista global, quer do desempenho percecionado por cada uma das classes de serviço.

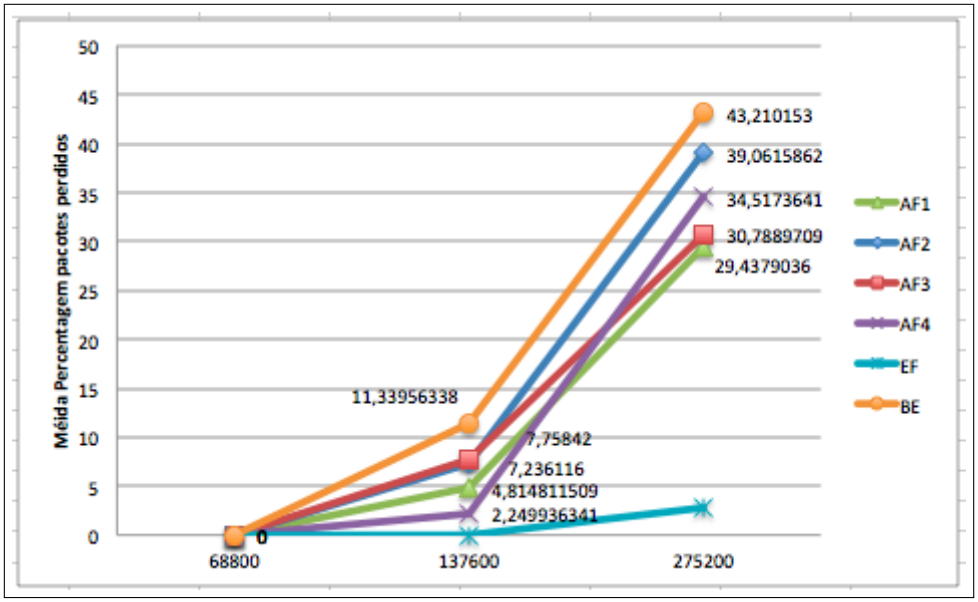
6.3.2 Cenário 2: Encaminhamento com Diferenciação nos encaminhadores da Rede

Para simular este cenário foi utilizado o modelo *DiffServ* de [1], passando desta forma a topologia *ISP* a ser encarada como um domínio de diferenciação, constituída pelos encaminhadores interiores e os encaminhadores de fronteira.

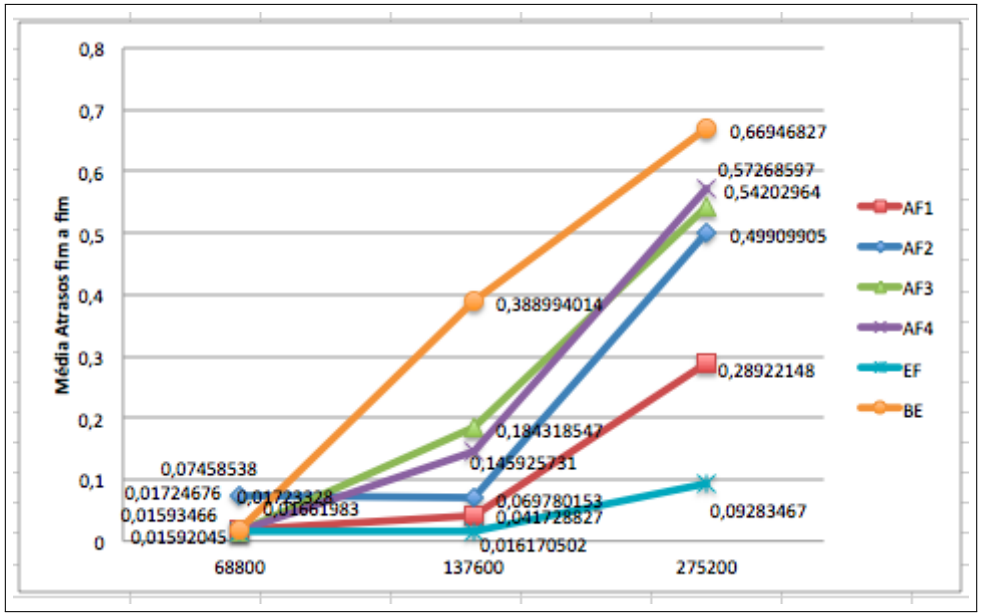
As filas definidas para os dispositivos de rede passaram a ser do tipo *DiffServ-Queue*, o que na prática correspondeu a criar seis filas distintas em cada dispositivo, uma para cada comportamento (*AF1*, *AF2*, *AF3*, *AF4*, *EF* e *BE*).

Para todos os cenários as filas têm a capacidade de 100 pacotes de dados por fila, e foram configuradas de forma a atribuir precedência absoluta à fila de tráfego *EF*, aplicando um algoritmo de escalonamento *Priority Queue* na sua saída, tal como foi descrito no funcionamento do modelo *DiffServ* na secção 5.2.1. Para definir diferentes níveis de descarte nas filas *AF*(*AF1, AF2, AF3 e AF4*) foi aplicado uma algoritmo *WRED*, conferindo maior prioridade à fila *AF1*, relativamente às restantes filas *AF*. A fila menos prioritária, a fila de tráfego *BE*, foi combinada com as filas *AF* através de um algoritmo *WRR*.

Tal como no cenário de encaminhamento sem QoS, foram realizados 3 conjuntos de 30 simulações independentes, com taxas de transmissão, por fluxo, distintas por conjunto. Considerando que neste cenário existe diferenciação de tráfego por classe de serviço, os resultados apresentados na Figura 6.3 correspondem aos valores médios de percentagem de pacotes perdidos e ao atraso fim a fim com percentil de 50, de cada uma das classes de serviço.



(a) Média da percentagem de pacotes perdidos por classe



(b) Média do atraso fim a fim por classe

Figura 6.3: Resultados obtidos no cenário de diferenciação nos encaminhadores

Pela Figura 6.3 podemos verificar que existe uma classe de serviço com um desempenho acima da média, no que refere a percentagem de pacotes perdidos e valor de atraso fim a fim. Trata-se da classe de serviço mais prioritária nas filas dos encaminhadores da rede, a classe *EF*. As restantes classes apresentam um desempenho tanto melhor quanto a prioridade que têm nas filas, sendo que a classe *BE* é a que pior desempenho apresenta.

À medida que a carga de tráfego na rede aumentou o desempenho de todas as

classes tendeu também a piorar, no entanto, verificamos que a classe *EF* é sempre a menos afetada. Claramente a estratégia de diferenciação ao nível das filas dos encaminhadores da rede funciona em pleno, de acordo com as configurações efetuadas.

Nesta estratégia o caminho escolhido pelos encaminhadores para encaminhar os pacotes de dados é obtido independentemente da classe de serviço, e baseia-se, tal como no cenário sem diferenciação, apenas no caminho mais curto em número de saltos. Os pacotes de dados pertencentes a fluxos com o mesmo par origem/destino são encaminhados pelo mesmo caminho fim a fim.

6.3.3 Cenário 3: Encaminhamento com diferenciação ao nível da Rede

O terceiro cenário pressupôs a inclusão nos testes do modelo de encaminhamento por classes de tráfego. Os fluxos de tráfego são encaminhados na rede aplicando o algoritmo de caminho mais curto que utiliza métricas baseadas nos requisitos das aplicações, em vez do número de saltos.

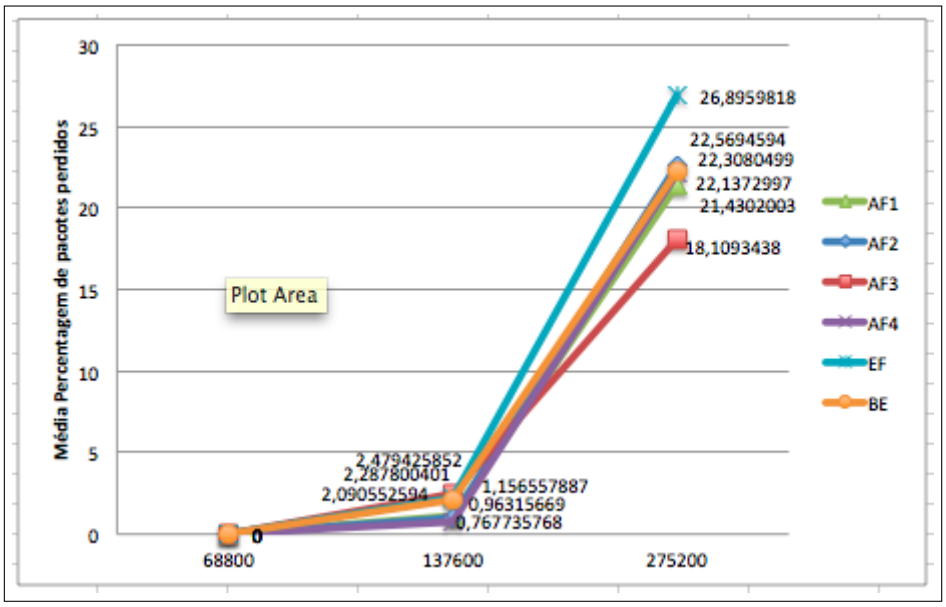
As filas nos dispositivos de rede continuam, tal como no cenário 2, a ser do tipo *DiffServ Queue*, mas sem qualquer tipo de diferenciação, ou seja, as seis filas de tráfego existem, mas com políticas de descarte e nível de precedência idênticos. O objetivo passou por testar o desempenho de um modelo onde a diferenciação do tráfego apenas é efetuada ao nível do encaminhamento.

O acondicionamento do tráfego para seis filas distintas, apesar de não conferir qualquer prioridade entre classes, permitiu avaliar o estado da rede da perspectiva de cada uma das classes de serviço. Com os valores observados nas filas dos dispositivos, em termos de atrasos e perdas de pacotes, passou a ser possível definir um caminho de custo mínimo por classe de serviço para cada destino, neste caso, seis caminhos por destino.

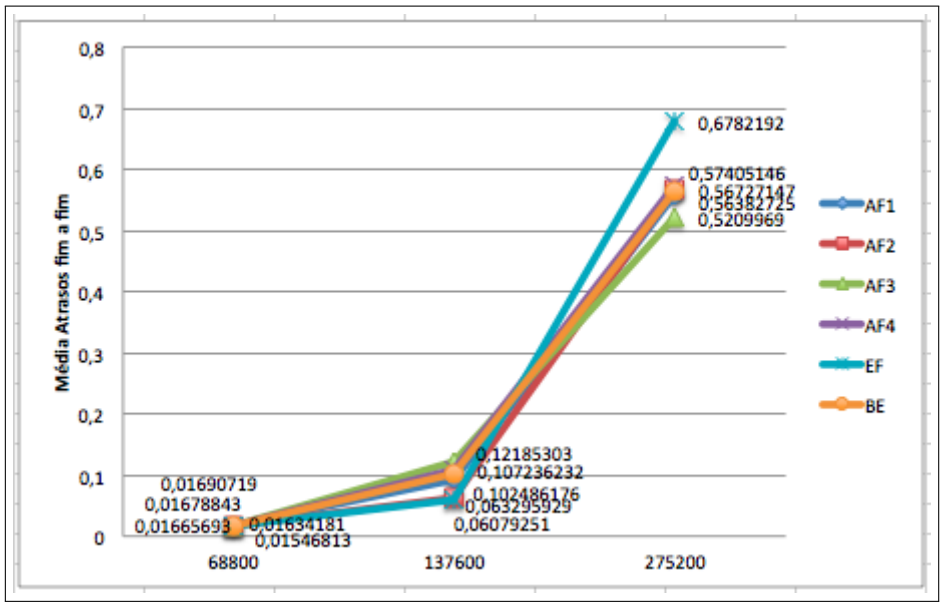
O tempo de monitorização das filas definido para este cenário foi de *2seg* e mais uma vez foram gerados 15 fluxos de tráfego *CBR* com pares origem/destino escolhidos aleatoriamente do grupo de encaminhadores de fronteira.

As métricas avaliadas no processo de monitorização dependeram, tal como foi referido no capítulo de implementação, da classe de serviço. No caso das classes de tráfego *AF* foi avaliada a percentagem de pacotes perdidos nas filas, a classe *EF* foi avaliada pelo atraso nas filas e a classe BE não foi avaliada na monitorização, tendo sido o custo das ligações definido com um custo fixo de 1.

É de prever que, com o dinamismo desta abordagem, o desempenho, em termos de atraso médio fim a fim e de percentagem de pacotes perdidos, tenha uma melhoria significativa relativamente aos modelos de encaminhamento sem diferenciação. Na Figuras 6.4 estão representados os gráficos com as médias dos parâmetros de desempenho, obtidas no final dos 3 conjuntos de simulações, de cada classe de serviço.



(a) Média da percentagem de pacotes perdidos por classe



(b) Média do atraso fim a fim por classe

Figura 6.4: Resultados obtidos no cenário de diferenciação ao nível do encaminhamento

Nas imagens (a) e (b) da Figura 6.4 verificamos que o desempenho das classes é bastante semelhante, e diminuiu à medida que aumentou a carga de tráfego na rede. Estes resultados tão aproximados entre classes de serviço, são justificados pelo facto de que os pacotes de dados não sofrem nenhum tratamento diferenciado nos encaminhadores, e porque é gerada, em média, a mesma quantidade de tráfego para todas as classes de serviço.

Nesta estratégia, os pacotes de dados são encaminhados por rotas contidas nas tabelas de encaminhamento, de acordo com a classe de serviço a que pertencem. Entre os mesmos pares origem/destino podem existir caminhos mais curtos distintos, dependendo do estado da rede percecionado por cada uma das classes de serviço. Ou seja, o melhor caminho para determinado destino baseado na perceção de uma das classes de serviço não é necessariamente o mesmo para outra classe distinta. Ao analisar os ficheiros gerados no final da simulação pela classe *StatCollector*, podemos verificar, especificamente no ficheiro dos registos de entrada e saída dos pacotes das filas, exatamente em que filas dos encaminhadores os fluxos de tráfego passaram, e determinar os caminhos escolhidos por cada classe de serviço.

Tal como era pretendido, o valor médio de atraso das 30 simulações para a classe de serviço *EF* foi o mais baixo, para situações de carga média, e o valor médio da percentagem de pacotes perdidos para as classes de tráfego AF também apresentou os valores menores. Isto significa que o objetivo de otimizar o desempenho das classes de acordo com os requisitos das mesmas é viável, para situações em que a rede não tenha uma carga de tráfego muito elevada.

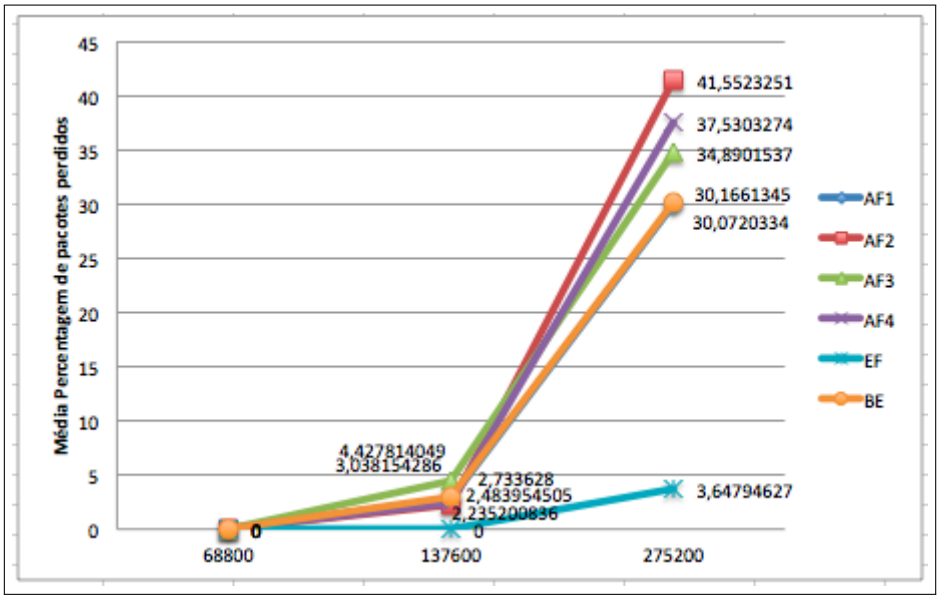
6.3.4 Cenário 4: Encaminhamento com diferenciação nos encaminhadores e ao nível da Rede

O último cenário alvo de testes apresenta a estratégia proposta no âmbito desta dissertação, onde é aliado o modelo de diferenciação nos encaminhadores do cenário 2 com o modelo de encaminhamento por classes de serviço do cenário 3.

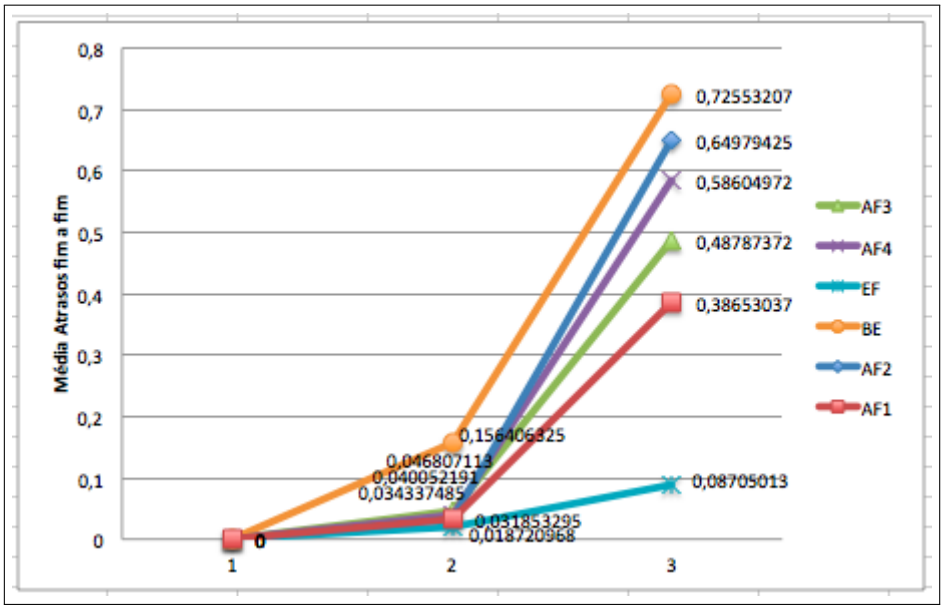
As filas *DiffServ* dos dispositivos de rede dos encaminhadores foram configuradas para efetuar a diferenciação dos pacotes de dados por classes de serviço, escalonando os pacotes para as respetivas filas de comportamento associado à classe. Foi definido o período de monitorização das filas para *2s* e, tal como o cenário 3, aplicado o protocolo de encaminhamento por classes de serviço.

O objetivo deste último cenário era determinar em que medida é melhorado o

desempenho da rede conciliando os dois modelos de diferenciação. Os resultados médios dos parâmetros de desempenho podem ser visualizados na Figura 6.5.



(a) Média da percentagem de pacotes perdidos por classe



(b) Média do atraso fim a fim por classe

Figura 6.5: Resultados obtidos no cenário Encaminhamento por Classes de Serviço

Na Figura 6.5 (a) e (b) verificamos que a classe de serviço *EF* obtém o melhor desempenho relativamente às restantes classes, pelo facto de ocorrer diferenciação nas filas dos encaminhadores, e esta classe ser a de maior prioridade. Quando a carga da rede é reduzida, esta classe apresenta uma percentagem de pacotes perdidos média praticamente nula, e um atraso médio também muito baixo, na ordem do atraso do

canal de comunicação (*2ms*).

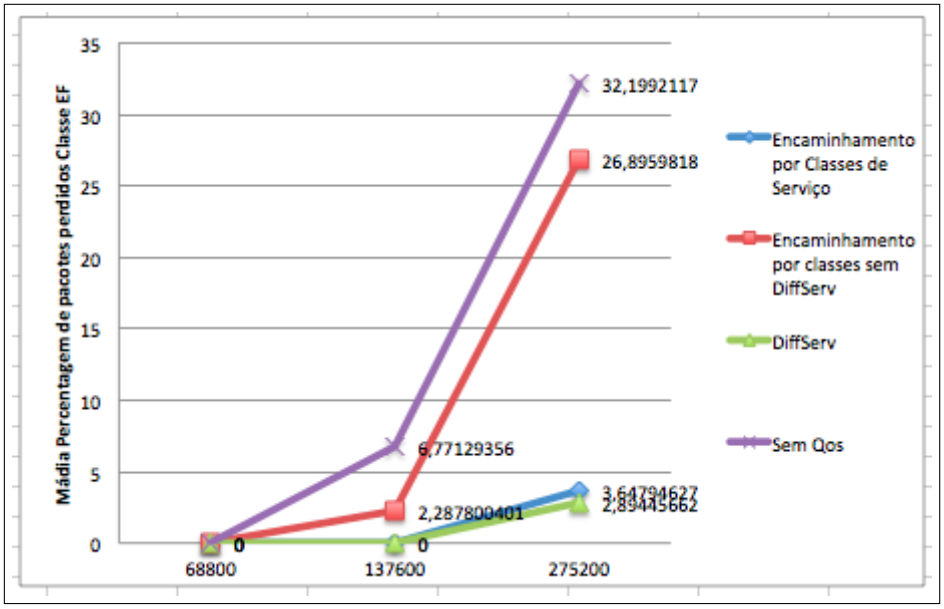
As classes de tráfego *AF* apresentam também um bom desempenho, no sentido que o parâmetro de referência para as aplicações que geram este tipo de tráfego, a percentagem de pacotes perdidos, se mantém na ordem dos *2%* apenas.

A classe de serviço *BE* apresenta o pior desempenho a nível de atraso médio, pelo facto de ser a classe de menor prioridade nas filas de diferenciação dos encaminhadores. No entanto, apresenta, para uma carga de rede reduzida, uma baixa percentagem de pacotes perdidos. Todas as classes perdem em desempenho à medida que a carga de tráfego a circular na rede aumenta, continuando a classe *EF* a sair claramente privilegiada com esta estratégia.

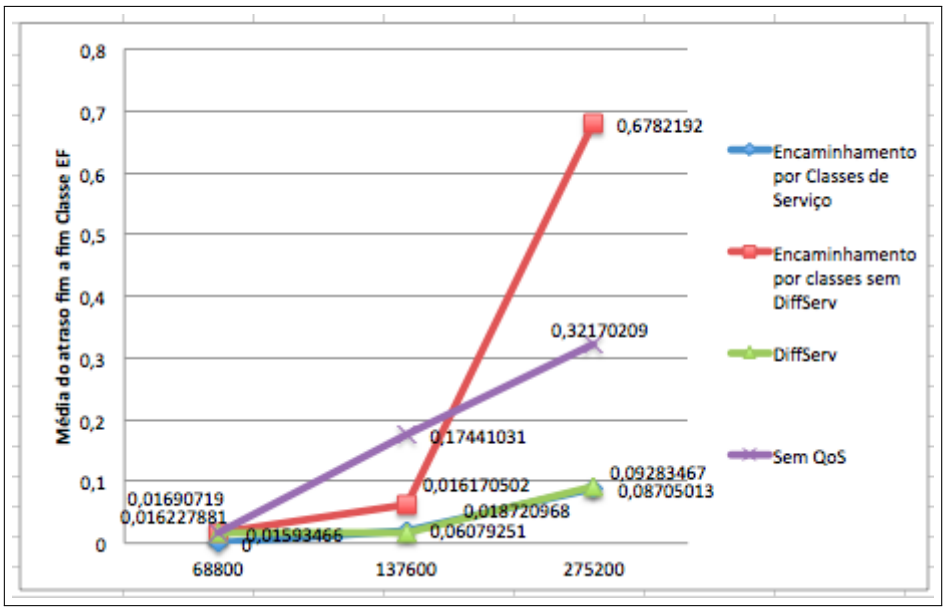
6.3.5 Análise dos resultados

Apesar dos resultados observados nos 4 cenários de testes estarem em concordância com as expectativas, o objetivo principal deste capítulo de testes passa por aferir o real benefício de optar por uma estratégia de encaminhamento por classes de serviço em alternativa à estratégia tradicional de melhor esforço, em vigor na *Internet*. Neste ponto, será efetuada uma análise comparativa dos 4 modelos de encaminhamento, para determinar se a estratégia proposta no âmbito da dissertação apresenta um melhoria significativa nos parâmetros de desempenho de cada classe de serviço.

Nas Figuras 6.6, 6.7 e 6.8 são apresentados os resultados das simulações dos 4 cenários de testes, para as classes de serviço *EF*, *AF1* e *BE*, respetivamente. Os resultados das classes *AF2, AF3 e AF4* foram omitidos nesta discussão crítica, já que se tratam de classes de serviço com requisitos em termos de percentagem de pacotes perdidos, tal como a *AF1*, já discutidos nos cenários apresentados.

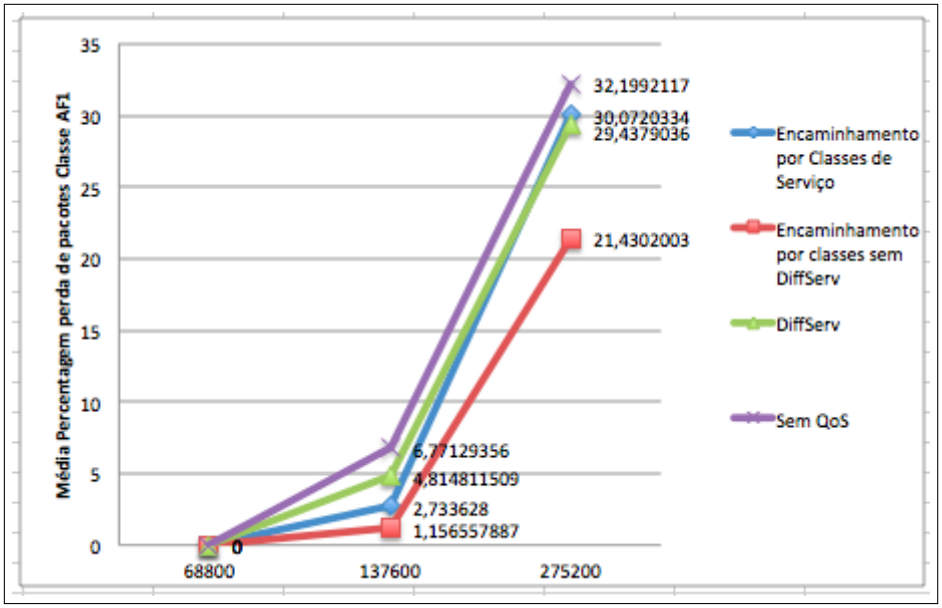


(a) Classe EF - Percentagem de pacotes perdidos

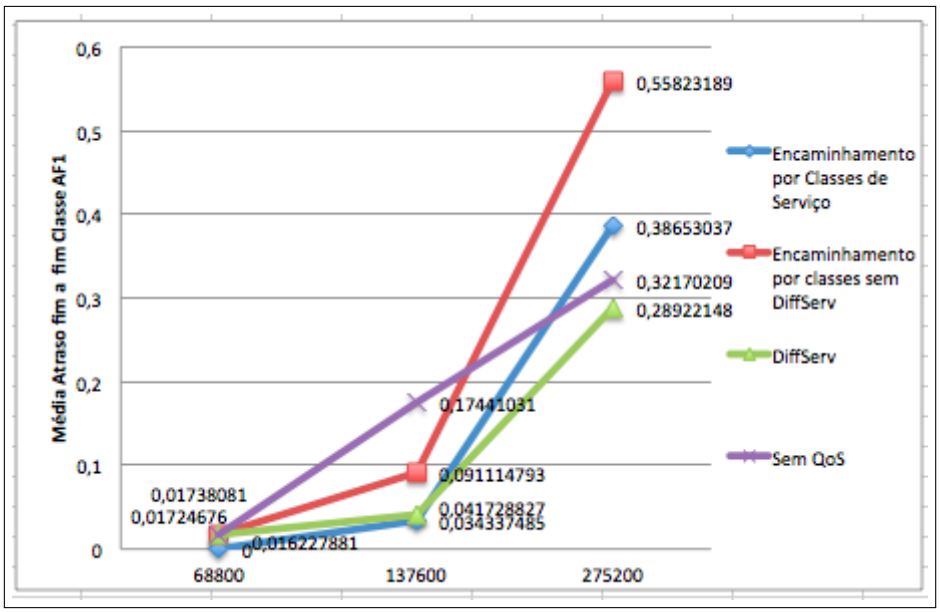


(b) Classe EF - Atraso fim a fim

Figura 6.6: Resultados da Classe EF nos 4 cenários de encaminhamento

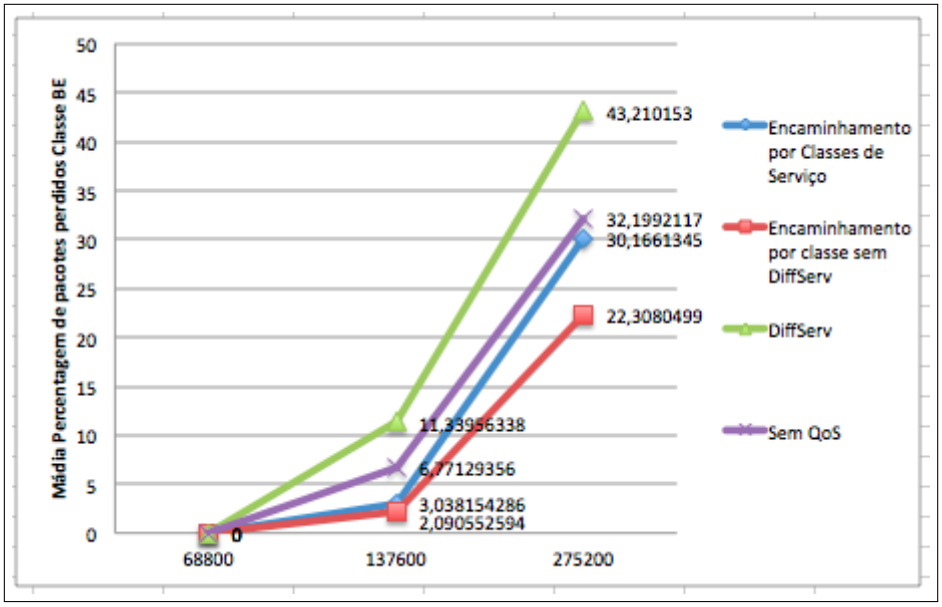


(a) Classe AF1 - Percentagem de pacotes perdidos

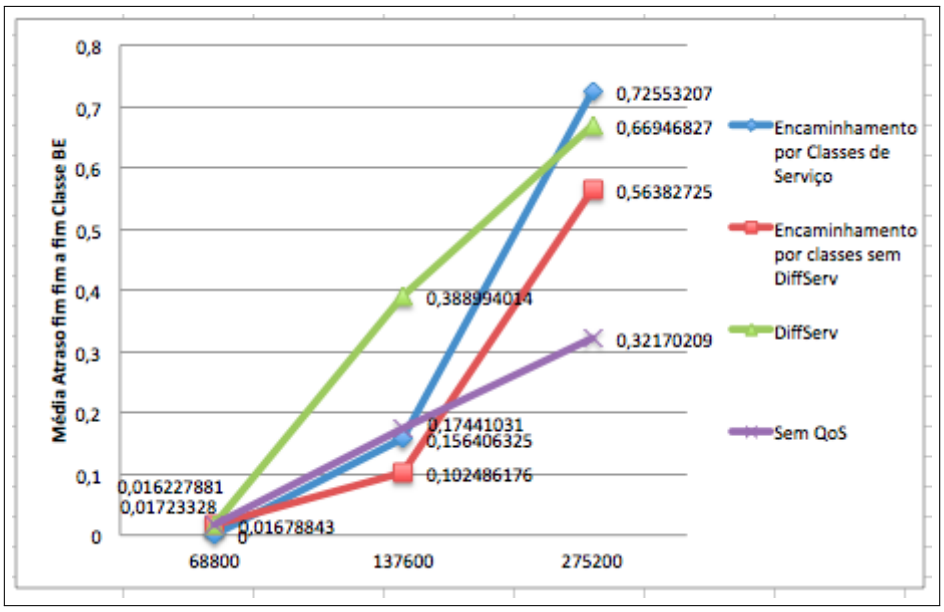


(b) Classe AF1 - Atraso fim a fim

Figura 6.7: Resultados da Classe AF1 nos 4 cenários de encaminhamento



(a) Classe BE - Percentagem de pacotes perdidos



(b) Classe BE - Atraso fim a fim

Figura 6.8: Resultados da Classe BE nos 4 cenários de encaminhamento

Ao analisar os gráficos com os resultados das médias calculadas, apresentados por classe de serviço para cada um dos cenários de encaminhamento, verificamos que, nas Figuras 6.6(a) e 6.6(b), onde é representado o resultado do desempenho da classe *EF*, uma estratégia com diferenciação de tráfego, seja nas filas dos encaminhadores ou ao nível do encaminhamento, apresenta um desempenho superior à estratégia sem diferenciação, designada neste contexto de *"Sem QoS"*. Para situações de muita carga de tráfego na rede, uma estratégia de encaminhamento por classes sem diferenciação nas filas dos encaminhadores, não é claramente suficiente otimizar o atraso fim a fim

da classe de serviço *EF*.

Nas Figuras 6.7(a) e 6.7(b) é representado o desempenho da classe *AF1*, uma classe de serviço que engloba os fluxos de dados das aplicações sensíveis a perda de pacotes, e verificamos que qualquer uma das estratégias de encaminhamento com diferenciação apresenta um bom desempenho, mesmo para cargas de tráfego elevadas. No entanto, não são observadas melhorias muito significativas em relação ao protocolo de encaminhamento tradicional.

Nas Figuras 6.8(a) e 6.8(b) é representado o desempenho da classe *BE*, a classe de serviço sem requisitos de *QoS*. Para esta classe o caminho mais curto é calculado em função do número de saltos, e cada ligação, neste contexto de simulação, tem permanentemente o custo de 1. O desempenho desta classe é significativamente afetado quando é aplicada uma estratégia de encaminhamento com diferenciação apenas ao nível das filas dos encaminhadores, pelo facto de se tratar da classe menos prioritária, e de todos os fluxos percorrerem o mesmo caminho entre pares origem/destino iguais, sem considerar a classe de serviço a que pertencem. Com o aumento da carga de tráfego na rede o desempenho da classe *BE* nas 2 estratégias de diferenciação ao nível do encaminhamento tende a piorar, relativamente à estratégia *"Sem QoS"*.

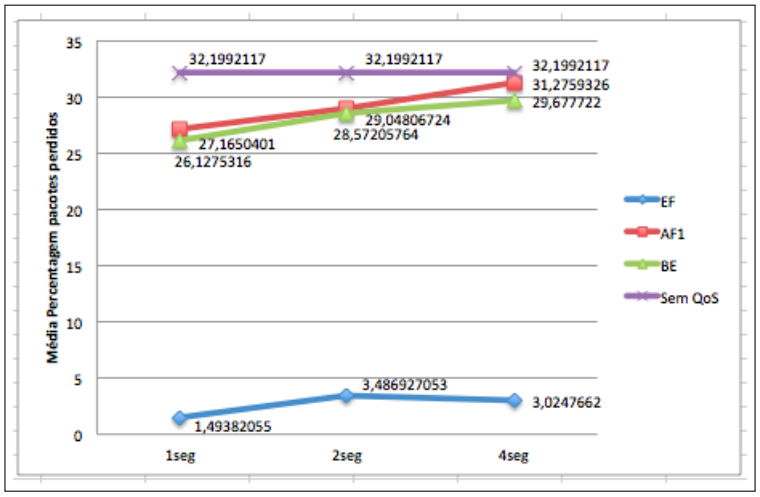
De uma forma global, ao analisar o desempenho das classes nos diferentes cenários, verificamos que efetivamente a estratégia de diferenciação nos encaminhadores confere um excelente desempenho às classes de *QoS*, no entanto, prejudica muito o desempenho da classe *BE*. A estratégia de encaminhamento com diferenciação ao nível do encaminhamento apresenta bom desempenho mas, de uma forma geral, apenas em situações de pouca carga, e não trata com mais prioridade as classes de *QoS*. Por sua vez, a estratégia de encaminhamento por classes de serviço com diferenciação nas filas dos encaminhadores da rede apresenta o desempenho desejado. As classes de *QoS* são tratadas de forma prioritária e, mesmo para situações de muita carga de tráfego na rede, não afetam muito significativamente o desempenho da classe *BE*.

Alteração do período de monitorização

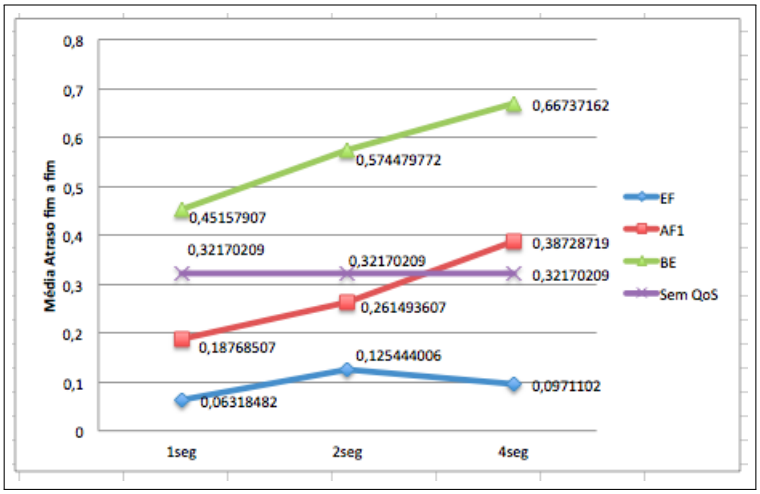
Devemos considerar que neste tipo de cenários, em que as atualizações dos custos das ligações são frequentes durante as simulações, possa ocorrer a perda de pacotes de dados durante o próprio processo de mudança de rota e mesmo algum aumento no atraso fim a fim, não podendo este fator condicionar o desempenho das classes de *QoS*. A escolha do período de monitorização deve ser bem ponderada. A escolha deste parâmetro deve ser baseada na quantidade e tipo de tráfego que vai circular na rede, correndo o risco de, como já foi referido no capítulo da estratégia proposta,

prejudicar o desempenho da estratégia com demasiadas atualizações das tabelas de encaminhamento, ou aproximar a estratégia do modelo de melhor esforço com um número insuficiente de atualizações.

Neste ponto serão apresentados graficamente os resultados da alteração do tempo de monitorização na estratégia de encaminhamento por classes de serviço proposta. Foram repetidas as 30 simulações para mais 2 cenários, monitorização a cada 1 segundo e a cada 4 segundos, para uma taxa de transmissão de *275200bps*. Na Figura 6.9 podemos observar o resultado obtido para as classes de tráfego *AF1, EF e BE*, e como termo de comparação são também considerados os resultados da estratégia *Sem QoS*.



(a) Média da percentagem de pacotes perdidos por classe



(b) Média do atraso fim a fim por classe

Figura 6.9: Resultados obtidos com alteração do tempo de monitorização

Ao analisar os gráficos apresentados podemos constatar a relevância da escolha do tempo de monitorização. O desempenho das classes de tráfego é afetado significativamente com a mudança deste parâmetro.

Os valores apresentados refletem a média das 30 simulações, sendo que a Figura 6.9(a) mostra a média da percentagem de pacotes perdidos das classes *AF1*, *EF*, *BE* e, como referência, os valores obtidos nas simulações do cenário de encaminhamento sem Qualidade de Serviço. A percentagem de pacotes perdidos aumenta em todas as classes à medida que aumenta o tempo de monitorização e, no caso das classes menos prioritárias, aproxima-se do modelo de melhor esforço (*Sem QoS*). A classe *EF* continua a apresentar um desempenho muito acima da média, isto porque continua a ser a classe mais prioritária dentro das filas dos encaminhadores, servida por um algoritmo de escalonamento *Priority Queue*.

Na Figura 6.9(b) estão representados os valores médios de atraso fim a fim com percentil de 50. Tal como aconteceu com os valores da percentagem de pacotes perdidos, no mesmo contexto de simulação, quanto maior o período de monitorização, maior o atraso fim a fim. Ao realizarmos menos monitorizações durante a simulação, reduzimos a probabilidade de ocorrerem atualizações às tabelas de encaminhamento, o que consequentemente reduz o número de mudança de rotas no processo de encaminhamento. Com menos mudanças de rota é de esperar que o valor de atraso fim a fim, percecionado pelas diferentes classes de tráfego, aumente em consequência do processo de diferenciação de que são alvos nas filas dos encaminhadores.

Capítulo 7

Conclusão

Neste capítulo é realizada uma síntese do trabalho realizado ao longo da dissertação. É efetuada uma reflexão acerca das motivações para a elaboração do modelo de encaminhamento proposto e identificados os principais objetivos. São também alvo de análise crítica os resultados obtidos na simulação da estratégia, e apresentadas algumas considerações para trabalho futuro.

7.1 Conclusão

Todo o trabalho realizado ao longo da presente dissertação prende-se essencialmente com dois conceitos: encaminhamento em redes *IP* e qualidade de serviço. A introdução de qualidade de serviço no processo de encaminhamento em redes *IP* não é uma tarefa trivial, e, por esse facto, o tema tem servido de motivação para a realização de diversos estudos na área das redes de computadores.

O modelo de encaminhamento atual não é claramente suficiente para atender às exigências das aplicações emergentes. É necessário influenciar, de alguma forma, o processo de encaminhamento para que passe a considerar essas novas exigências. No entanto, ao considerar qualquer tipo de alteração ao modelo de encaminhamento da *Internet*, devemos primeiramente considerar a sua dimensão e a implicação que advém da mais pequena alteração. Uma nova abordagem deve ser aplicada coerentemente, não só no encaminhamento intra-domínio como também no encaminhamento inter-domínio, numa perspetiva fim a fim.

Os diversos estudos que têm surgido nos últimos anos apresentam diferentes estratégias mas, de forma geral, baseiam-se em duas abordagens: uma abordagem por fluxo e uma abordagem por classes de serviço. A abordagem por fluxo assenta no

princípio de que o processo de encaminhamento é adaptado para que os encaminhadores calcularem as melhores rotas por fluxo de dados. Os encaminhadores têm que manter permanentemente a informação de todos os fluxos de dados que percorrem as rotas onde estão inseridos. Esta abordagem apresenta por este facto problemas de escalabilidade. Na abordagem por classes de serviço, a qualidade se serviço é garantida no encaminhamento por classe de serviço e não por fluxo. Os pacotes dos fluxos de dados são marcados numa das classes de serviço e tratados pelos encaminhadores da rede de acordo com a classe a que pertencem. O número de classes de serviço é imensamente reduzido se pensarmos na quantidade de fluxos que circula na rede, sendo esta abordagem considerada muito mais escalável que a abordagem por fluxo.

O modelo *Diffserv* proposto pelo *IETF* permite que seja efetuada a diferenciação do tráfego nas filas dos encaminhadores, no entanto, não influência o processo de encaminhamento. O modelo permite que seja efetuada a diferenciação dos fluxos de tráfego por classes de serviço, mas o caminho por onde os fluxos percorrem a rede é independente da classe a que pertencem.

No processo de encaminhamento tradicional em redes *IP*, os encaminhadores trocam informação de conectividade e acessibilidade entre si para que sejam construídas as tabelas de encaminhamento. Cada encaminhador da rede possui uma tabela com os interfaces de saída adequados para direcionar os pacotes de dados de acordo com o destinatário. O protocolo de encaminhamento *OSPF* proposto pelo *IETF* é o mais amplamente utilizado nas redes de computadores reais. Por isso e pelos restantes motivos identificados, na tentativa de adaptar os mecanismos existentes ao propósito desta dissertação, o modelo *DiffServ* e o protocolo de encaminhamento *OSPF* foram a base de desenvolvimento da estratégia proposta no âmbito da dissertação: *Encaminhamento de tráfego por Classes de Serviço para redes DiffServ*.

Desta forma, considerando as motivações e os mecanismos supracitados, a estratégia foi implementada no simulador de redes *NS-3* e o seu desenvolvimento passou pelas seguintes etapas:

- Estudo e integração do modelo de diferenciação *DiffServ* para *NS-3*. Esta etapa teve uma especial importância no processo de implementação pois permitiu a familiarização com o ambiente de simulação escolhido, o *NS-3*. Com a integração deste modelo, passou a ser possível simular um domínio *DiffServ* com diferenciação dos pacotes de dados nas filas dos encaminhadores. O modelo permite distinguir entre seis classes de serviço, e para cada uma das classes é criada uma fila nos dispositivos de rede dos encaminhadores. Cada fila com

políticas de descarte e precedência configuráveis. O modelo permite também o estabelecimento de acordos entre o *ISP* e o cliente das aplicações, garantindo, desta forma, o relacionamento entre os requisitos de funcionamento das aplicações e o tratamento dado aos pacotes de dados no domínio de diferenciação.

- Definição das classes de serviço e dos parâmetros de desempenho associados. Foram nesta estratégia definidas 6 classes de serviço e respectivamente 6 filas: uma classe *EF* muito sensível a atrasos; quatro classes de serviço *AF* (*AF1, AF2, AF3 e AF4*) sensíveis a perdas de pacotes e, por último, uma classe *BE* sem requisitos de *QoS*. A fila mais prioritária é a *EF*, seguida da *AF1*, *AF2*, *AF3*, *AF4* e por último a de menor prioridade a *BE*. Nesta estratégia foi definido que o custo das ligações, percecionados por cada uma das classes de serviço, seria absolutamente independente e, baseado no desempenho de cada fila, observado num processo de monitorização periódico. Em cada monitorização são avaliados os parâmetros de desempenho associados a cada classe de serviço, atraso médio da fila para a classe *EF*, e percentagem de pacotes perdidos para cada uma das classes *AF*. Com os valores observados, por comparação com valores obtidos em monitorizações anteriores, é determinada a necessidade de alterar os custos das ligações, no sentido de as tornar mais ou menos atrativas, de acordo com o desempenho percecionado pelas classes.

- Estudo e adaptação do modelo de encaminhamento global *unicast* nativo ao *NS-3*. O modelo de encaminhamento global, conhecido no âmbito do simulador por *GOD*, é baseado no funcionamento do *OSPF*. Neste modelo as rotas inseridas nas tabelas de encaminhamento dos encaminhadores são calculadas a partir de uma métrica simples baseada no número de saltos. Foi necessário adaptar o protocolo para que utiliza-se métricas baseadas nos requisitos das aplicações. Os *LSAs* foram alterados para que passassem a incluir o custo do caminho por cada uma das classes de serviço, ou seja, cada ligação deixou de ter um custo único associado e passou a ter um custo associado por cada classe, relacionado com os requisitos de *QoS*. Para a classe *EF* o custo é obtido em função do atraso, para as classes *AF* em função da percentagem de pacotes perdidos, e em número de saltos para a classe *BE*. Com os novos custos passam a ser calculados os caminhos mais curtos por classe de serviço, caminhos esses, posteriormente inseridos nas tabelas de encaminhamento, também por classe de serviço. Com isto, quando um pacote de dados chega a um encaminhador, este verifica, pelo cabeçalho *IP*, a classe de serviço marcada no pacote e o destinatário, e escolhe a melhor rota na tabela de encaminhamento de acordo com os dois parâmetros, ou seja, a melhor rota para a classe de serviço específica.

Cumprindo o último objetivo, foi efetuada a comparação da estratégia de encaminhamento proposta com outras estratégia de encaminhamento, nomeadamente: encaminhamento sem diferenciação, encaminhamento com diferenciação nas filas dos nós e encaminhamento com diferenciação ao nível do encaminhamento apenas. Foi desenvolvido um cenário de testes com 21 encaminhadores (8 encaminhadores fronteira e 13 encaminhadores internos) com múltiplos canais de comunicação, no sentido de criar caminhos alternativos. Foram configuradas 15 aplicações geradoras de tráfego com origem e destino escolhidos de forma aleatória de entre os encaminhadores fronteira. As aplicações foram associadas, também de forma aleatória, às 6 classes de serviço, fazendo com que, em média, a distribuição do tráfego pelas 6 classe fosse uniforme. A distribuição uniforme de tráfego levou a que a quantidade de tráfego com *QoS*, (distribuído pelas classes *EF, AF1, AF2, AF3 e AF4*), fosse muito superior à quantidade de tráfego *BE*, criando um cenário muito adverso ao tráfego sem requisitos de *QoS*.

Mesmo nas condições apresentadas, com a análise dos resultados obtidos nas diferentes simulações, verificamos que a estratégia de encaminhamento, em que o modelo de diferenciação nas filas dos encaminhadores é conciliado com um protocolo de encaminhamento por classes de serviço, apresenta um desempenho superior ao modelo de encaminhamento sem mecanismos de diferenciação (sem *QoS*). No entanto, à medida que a quantidade de tráfego que circula na rede aumenta, para níveis de de congestão relacionados com os limites dos canais de comunicação, o desempenho tende a degradar-se.

O tráfego pertencente à classe de serviço *EF* é o que apresenta, no decorrer das simulações, o melhor desempenho, facto relacionado com a presença de uma *Priority Queue* a servir as filas dessa classe e, com o algoritmo de caminho mais curto, baseado no atraso observado nas filas. O objetivo de otimizar os caminhos em função do atraso, para este tipo de tráfego, é nitidamente alcançado.

O desempenho das classes de serviço *AF* apresenta também algumas melhorias, à primeira vista não muito significativas, mas, se considerar-mos que a quantidade de tráfego das classes de serviço *AF*, com requisitos relacionados com a percentagem de pacotes perdidos, gerado neste contexto de simulação, representa a maioria do tráfego, verificamos que as melhorias são muito relevantes quando comparamos o modelo proposto com o modelo sem mecanismos de diferenciação.

Relativamente a classe de tráfego *BE*, devemos primeiramente considerar que:

o tráfego desta classe recebe o tratamento de prioridade inferior nas filas dos encaminhadores, as rotas são estabelecidas a partir de uma métrica baseada no número de saltos, mantendo-se as rotas dos fluxos inalteradas durante toda a simulação e, o tráfego com requisitos de *QoS* representa a maioria absoluta do tráfego que percorre o domínio de diferenciação. No entanto, com esta estratégia, o desempenho da classe de serviço *BE* é também melhorado relativamente à estratégia sem *QoS*. O facto de o tráfego com requisitos de *QoS* ser encorajado a evitar caminhos congestionados, sendo os pacotes encaminhado por rotas alternativas, contribuí para o bom desempenho da classe *BE*, que acaba por, ao longo da simulação, utilizar uma rota quase dedicada ao seu tráfego. Apesar da perceção da rede ser independente entre as diferentes classes, naturalmente todo o tráfego gerado, independentemente dos requisitos de *QoS*, influência o desempenho observado por cada uma delas, quer em termos de atraso médio nas filas, quer em termos de percentagem de pacotes perdidos. O que significa que: por exemplo, a existência de fluxos de tráfego *BE* a percorrer determinados caminhos na rede, com este protocolo de encaminhamento, desencoraja os fluxos de tráfego com *QoS* de percorrerem os mesmos caminhos.

Por último, no sentido de verificar em que medida o período de monitorização das filas afeta o desempenho da estratégia, o cenário foi simulado com diferentes tempos de monitorização. Os resultados foram exatamente os expectados: quanto mais periódica for a monitorização, melhor o desempenho de cada uma das classes de serviço, de forma uniforme; Com a diminuição do período de monitorização, observamos um degradação do desempenho, aproximando o modelo cada vez mais do modelo de melhor esforço. O facto de serem realizadas mais monitorizações às filas, permite acompanhar, de forma mais permanente, o estado das filas, o que, tendêncialmente, leva a atualizações mais frequentes das tabelas de encaminhamento. Por este motivo, a escolha do período de monitorização deve ser bem ponderada durante a configuração da rede, correndo o risco de ganhar em desempenho mas, aumentar demasiado a sobrecarga introduzida pela troca de mensagens de controlo entre os encaminhadores (no processo de atualização das tabelas de encaminhamento).

Considerações para trabalho futuro

Infelizmente com o tempo disponível para a realização da dissertação, não me foi possível abordar, com o nível de detalhe que pretendia, todas as questões relacionadas com a problemática do encaminhamento por classes de serviço.

Um primeiro ponto a aprofundar seria a questão da diferenciação ao nível do

encaminhamento das classes de serviço *AF*. Apesar de, nesta estratégia, cada uma das classes *AF* ter uma perceção independente do estado da rede, a métrica utilizada no cálculo dos caminhos mais curtos é baseada, para as 4 classes, na percentagem de pacotes perdidos. Parece-me que seria importante refinar a escolha desta métrica para estas classes. No sentido de atribuir "sensibilidades" distintas entre as classes, à percentagem de pacotes perdidos observada.

Ainda relativamente à métrica relacionada com a percentagem de pacotes perdidos, que na estratégia proposta, foi obtida com base nos valores observados na monitorização, mas que não representa exatamente o valor obtido, mas sim é uma função do valor. Sendo esta uma métrica multiplicativa, para ser utilizada no cálculo do caminho mais curto, era necessário efetuar uma de duas opções: ou alterar o algoritmo de caminho mais curto, ou converter a métrica multiplicativa numa métrica aditiva (adequada ao *Dijkstra*).

Outra questão prende-se com a escolha do cenário de simulação, é fundamental realizar novos testes para comprovar a eficiência da estratégia num cenários de teste onde a quantidade de tráfego sem requisitos de *QoS* é superior à quantidade do tráfego das classes de *QoS*, à semelhança do que acontece, típicamente, nas redes de computadores reais.

Bibliografia

[1] S. Ramroop, "A diffserv model for the ns-3 simulator." http://www.eng.uwi.tt/depts/elec/staff/rvadams/sramroop/, 2010.

[2] J. Postel, "Internet Protocol." RFC 791 (INTERNET STANDARD), Sept. 1981. Updated by RFCs 1349, 2474, 6864.

[3] B. Deng, H. Liu, and M. Zheng, "Overview of qos routing."

[4] B. Carpenter, "Architectural Principles of the Internet." RFC 1958 (Informational), June 1996. Updated by RFC 3439.

[5] E. M. Schooler, "Qos in the internet: An overview," 1997.

[6] S. Keshav, *An Engineering approach to Computer Networking*. Corporate and Professional Publishing Group, May 15, 1997.

[7] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview." RFC 1633 (Informational), June 1994.

[8] S. Shenker and J. Wroclawski, "General Characterization Parameters for Integrated Service Network Elements." RFC 2215 (Proposed Standard), Sept. 1997.

[9] S. Shenker and J. Wroclawski, "Network Element Service Specification Template." RFC 2216 (Informational), Sept. 1997.

[10] J. Wroclawski, "Specification of the Controlled-Load Network Element Service." RFC 2211 (Proposed Standard), Sept. 1997.

[11] S. Shenker, C. Partridge, and R. Guerin, "Specification of Guaranteed Quality of Service." RFC 2212 (Proposed Standard), Sept. 1997.

[12] P. Almquist, "Type of Service in the Internet Protocol Suite." RFC 1349 (Proposed Standard), July 1992. Obsoleted by RFC 2474.

[13] S. Amante, B. Carpenter, S. Jiang, and J. Rajahalme, "IPv6 Flow Label Specification." RFC 6437 (Proposed Standard), Nov. 2011.

[14] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services." RFC 2475 (Informational), Dec. 1998. Updated by RFC 3260.

[15] K. Nichols, S. Blake, F. Baker, and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers." RFC 2474 (Proposed Standard), Dec. 1998. Updated by RFCs 3168, 3260.

[16] D. Grossman, "New Terminology and Clarifications for Diffserv." RFC 3260 (Informational), Apr. 2002.

[17] F. Baker, J. Polk, and M. Dolly, "A Differentiated Services Code Point (DSCP) for Capacity-Admitted Traffic." RFC 5865 (Proposed Standard), May 2010.

[18] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured Forwarding PHB Group." RFC 2597 (Proposed Standard), June 1999. Updated by RFC 3260.

[19] B. Davie, A. Charny, J. Bennet, K. Benson, J. L. Boudec, W. Courtney, S. Davari, V. Firoiu, and D. Stiliadis, "An Expedited Forwarding PHB (Per-Hop Behavior)." RFC 3246 (Proposed Standard), Mar. 2002.

[20] G. Malkin, "RIP Version 2." RFC 2453 (INTERNET STANDARD), Nov. 1998. Updated by RFC 4822.

[21] J. Moy, "OSPF Version 2." RFC 2328 (INTERNET STANDARD), Apr. 1998. Updated by RFCs 5709, 6549, 6845, 6860.

[22] D. Waitzman, C. Partridge, and S. Deering, "Distance Vector Multicast Routing Protocol." RFC 1075 (Experimental), Nov. 1988.

[23] J. Moy, "Multicast Extensions to OSPF." RFC 1584 (Historic), Mar. 1994.

[24] Q. Ma and P. Steenkiste, "Supporting dynamic inter-class resource sharing: A multi-class qos routing algorithm," in *In IEEE Infocom*, pp. 649–660, 1999.

[25] W. Zhou, P. Zhang, X. Bai, and R. Kantola, "A qos based routing algorithm for multi-class optimization in diffserv networks 1."

[26] H. Kochkar, T. Ikenaga, and Y. Oie, "Oie y, multi-class qos routing strategies based on the network state," *Computer Communications*, 2005.

[27] H. Kochkar, T. Ikenaga, and Y. Oie, "Qos routing algorithm based on multiclasses traffic load," in *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, vol. 4, pp. 2193–2198 vol.4, 2001.

[28] R. Technology, "Opnet- network planning simulation." http://www.riverbed.com/products-solutions/products/network-performance-management/network-planning-simulation/, 2013.

[29] I. SCALABLE Network Technologies, "Qualnet." http://web.scalable-networks.com/content/qualnet, 2013.

[30] S. Mccanne, S. Floyd, and K. Fall, "The network simulator ns-2." http://www.isi.edu/nsnam/ns/, 2011.

[31] T. Henderson, S. Roy, S. Floyd, and G. Riley, "Network simulator 3 ns-3." http://www.nsnam.org/, 2013.

[32] O. Community, "Network simulation framework omnet++." http://www.omnetpp.org/, 2013.

[33] Nsnam, "Ns3installation." http://www.nsnam.org/wiki/index.php/Installation, 2013.

[34] A. Inc., "Mac os x lion." www.apple.com, 2013.

[35] I. Free Software Foundation, "Gcc, the gnu compiler collection." http://gcc.gnu.org/, 2013.

[36] ns developers, "ns-3 allinone scripts." http://code.nsnam.org/ns-3-allinone, 2013.

[37] P. Jakma, V. Jardin, D. Lamparter, A. Schorr, D. Tejblum, and G. Troxel, "Quagga routing software suite." http://www.quagga.net/, 2012.